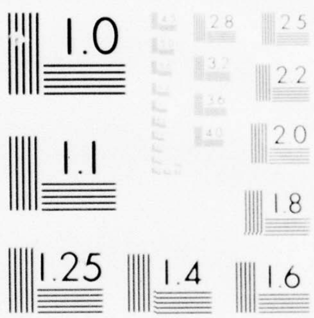
AD-A044 442 BROWN UNIV PROVIDENCE R I DIV OF APPLIED MATHEMATICS F/G 9/2
ABDUCTION ALGORITHMS FOR GRAMMAR DISCOVERY.(U)
JUN 77 S SHRIER N00014-77-C-0248

UNCLASSIFIED NL

1 OF 2
AD
A044442


1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

AD A044442

(12)

DEPARTMENT OF THE NAVY
Office of Naval Research
Contract N00014-77-C-0248

ABDUCTION ALGORITHMS FOR GRAMMAR DISCOVERY

by

Stefan Shrier

Division of Applied Mathematics
Brown University
Providence, R.I. 02912

June 1977

DEPARTMENT OF THE NAVY

Office of Naval Research

Contract N00014-77-C-0248

15

6

ABDUCTION ALGORITHMS FOR GRAMMAR DISCOVERY,

by

10

Stefan Shrier

9 Technical rept.,

Division of Applied Mathematics
Brown University
Providence, R.I. 02912

12 152p.

11 June 1977

065 300

mt

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER Unnumbered	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Abduction Algorithms for Grammar Discovery		5. TYPE OF REPORT & PERIOD COVERED Technical
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Stefan Shrier		8. CONTRACT OR GRANT NUMBER(s) N00014-77-C-0248 new
9. PERFORMING ORGANIZATION NAME AND ADDRESS Division of Applied Mathematics Brown University Providence, R. I. 02912		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Mathematical and Computer Sciences Div. Arlington, VA 2221		12. REPORT DATE June 1, 1977
		13. NUMBER OF PAGES 142
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Office of Naval Research 495 Summer Street Boston, Mass. 02210		15. SECURITY CLASS. (of this report)
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

See Distribution List at end of this report.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Grammatical Inference; Language acquisition; Syntax-controlled probability grammar; pattern theory; automata.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A model of language syntax acquisition is formulated as an inference problem: to guess the wiring diagram of an unreliable automaton. A constructive method is developed which solves the grammatical inference problem via an abductory inductive process applied to the sample strings generated by the stochastic automaton whose internal wiring diagram is unavailable for inspection. The right invariant equivalence classes which → next page

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. (contd.)

correspond to the states of the sought-for automaton are established by the training sequence and a teacher. The structural description of strings is found directly without a priori assumptions on the number of states (or lengths of strings). Several examples are used to illustrate the behavior of various algorithms which were developed to carry out the synthesis; among these is a fragment of English grammar. The same method for solution of the above problem can be used to establish word classes. The dictionary is partitioned into the classes induced by an equivalence relation defined by grammatical substitutability of words. All the algorithms are formally described and implemented on a digital computer in A Programming Language (APL). After some finite time, the algorithm establishes the minimal state, completely specified, deterministic, performance model automaton. The original grammar is obtained together with the frequency-defined probabilities which approximate the probabilities imposed on the rules of the grammar. That is, the states, wires, and approximate transition probabilities of the original automaton are obtained. These "abduction machines"are analyzed for convergence rates and robustness (stability with respect to an imperfect teacher).

Abstract

A model of language syntax acquisition is formulated as an inference problem: to guess the wiring diagram of an unreliable automaton. A constructive method is developed which solves the grammatical inference problem via an abductory inductive process applied to the sample strings generated by the stochastic automaton whose internal wiring diagram is unavailable for inspection. The right invariant equivalence classes which correspond to the states of the sought-for automaton are established by the training sequence and a teacher. The structural description of strings is found directly without a priori assumptions on the number of states (or lengths of strings). Several examples are used to illustrate the behavior of various algorithms which were developed to carry out the synthesis, among these is a fragment of English grammar. The same method for solution of the above problem can be used to establish word classes. The dictionary is partitioned into the classes induced by an equivalence relation defined by grammatical substitutability of words. All the algorithms are formally described and implemented on a digital computer in A Programming Language (APL). After some finite time, the algorithm establishes the minimal state, completely specified, deterministic, performance model automaton. The original grammar is obtained together with the frequency-defined probabilities which approximate the

probabilities imposed on the rules of the grammar. That is, the states, wires, and approximate transition probabilities of the original automaton are obtained. These "abduction machines" are analyzed for convergence rates and robustness (stability with respect to an imperfect teacher).

TABLE OF CONTENTS

ILLUSTRATIONS

TABLES

PREFACE

This dissertation pursues one of the tangential paths created by the discussions among Professors Leon Cooper (Physics), Walter Freiberger (Applied Mathematics), Ulf Grenander (Applied Mathematics), and Henry Kucera (Linguistics) which had been ongoing for several years. I am most grateful to my mentor, Professor Grenander, for posing the problem to me and for invaluable help, support and constructive criticisms during the development of this work. I should like to thank Professor Donald E. McClure and Professor Kucera for many suggestions, patience, and for serving as readers. I am grateful to Professor Freiberger for arranging my involvement with the project.

I also acknowledge Mr. R. J. Solomonoff for making available useful offprints and difficult-to-get primary reference material; Dr. Richard A. Vitale for his tutorial on basic probability theory; and Dr. Charles M. Strauss for personal support and friendship.

I am grateful for the National Defense Education Act Fellowship which supported me at Brown 1966-1969.

To Mrs. Ezoura Fonseca, for her good karma, expert typing, and flexibility, many thanks.

Chapter 1
Introduction

This paper deals with a model of language syntax acquisition. It is assumed that the artificial language has a finite description which we hope to discover on the basis of a finite sample of sentences. Specifically excluded is the simple formulation of the observations actually made, although a naive description is highly desirable. In the terminology of learning models, an insightful model is to be preferred over the rote learning exemplified in a list.

Proposal for the study of this problem was posed in the paper "Pattern Conception" by Miller and Chomsky [1957]; this paper elaborated on the virtues of finite state automaton models. An early description of a machine to carry out grammar discovery is given by Solomonoff in [1957]. Variations of this problem are found in artificial intelligence, human cognitive studies, pattern recognition, linguistics, and in systems theory under labels such as inductive inference, automaton identification and grammatical inference. The fine exposition by Fu [1974] in a chapter entitled "Grammatical Inference for Syntactic Pattern Recognition" surveys many approaches and also contains 49 references. Additional references can be found in Trakhtenbrot and Barzdin [1973]. More recent references are Adleman and Blum [1975] which deals with degrees of unsolvability of inductive inference problems and Angluin [1976] which explores complexity of the inference of finite state grammars from a finite set of positive and negative sample strings.

The following investigation is motivated toward the presentation of natural models. The investigation follows the paths initiated by Grenander [1974] in his abductory induction models. If we restrict consideration to regular grammars, the number of possible grammars is overwhelmingly large for even small size alphabets and modest numbers of variables. This implies that models for analogy to real world phenomena which exhibit language acquisition ability cannot be based on enumerative inference or finite search techniques; for this reason, we also seek an alternative to direct implementations of maximum likelihood solutions which select one grammar over another on probabilistic criteria by the solution of large scale linear systems.

In particular, algorithms are presented which carry out grammar discovery by the construction of the (right invariant) equivalence classes induced by the finite state automaton. The classes are established by means of a training sequence and a teacher. The number of possible partitions induced by an equivalence relation defined on a finite set of elements is known to be given by sums of Stirling numbers of the second kind. To reduce the combinatorial complexity of the problem, another equivalence relation is defined on the word dictionary in a natural way by the grammatical substitutability of words. It is used to partition this dictionary into grammatical equivalence classes. The prototype of each word class, that is, the representor of each of the established classes, is then used to carry out the synthesis of the minimal state automaton. The algorithms are imbedded in a statistical environment; they are studied experimentally and theoretically. Special attention is focussed on the

flexibility of the algorithms to accommodate non-systematic errors (e.g. a teacher which occasionally makes errors).

The model consists of two components: the first component is a teacher in a dual role. The teacher acts as a generative grammar which produces strings in a syntax-controlled probability language (Grenander [1967]). The random structure of the language is induced by imposing probabilities on the rules of the grammar. In this way, the production of some strings can be inhibited while the production of others can be made more likely. This adds to the teacher's grammar a facet of linguistic performance although it does not increase the generative power of the grammar. The teacher also serves as an acceptor of strings; it can judge whether or not a sentence presented to it by the learner is within its competence.

The second component, called the learner, consists of two principal procedures which carry out the construction of a copy of the teacher. The first procedure or phase is devoted to the classification of words into equivalence classes on the basis of grammatical substitutability; this procedure is in principle infinitary. These classes are familiar in structural linguistics where they are called families (Kulagina [1958]) or categories (Miller and Chomsky [1963]) although we do not use these classes in quite the same way. They are introduced here to reduce the combinatorial complexity of the constructions.

The first phase begins with the assumption that all words are in one equivalence class: this is the <u>tabula</u> <u>rasa</u> with which

the learner begins. The discovery of the classes is carried out by the resolution of the dictionary into the classes which form the required partition. The teacher randomly generates strings which are presented to the learner as a training sequence. For each string in the training sequence, a word is selected according to a weighted probabilistic strategy; this might be thought of as an attention function. This string is used to either strengthen the learner's belief about the selected word's membership in its present grammatical class or it is used to introduce a new hypothesis to be entertained about the relation of this word to the sought for partition of the dictionary. The term abduction, introduced by C.S. Peirce [1931] to describe the starting of a hypothesis, is applied to describe this process which either changes the class membership of the selected word or forms a new class with this selected word. All classes formed are characterized by a fixed representative word called a prototype.

The second phase carries out the discovery of the syntactic variables and the rewrite rules which govern these variables. Initially in this phase, the learner has a *tabula rasa* with respect to variables; the discovery of variables proceeds in a manner analogous to the phase one process. Each string in the training sequence is analyzed to determine initial string equivalence classes, i.e. the syntactic variables. Each string can be decoded with respect to the word class partition determined in phase one as described above. This indexing scheme is used to implement efficiently the process which determines the

partition of initial strings into the sought for syntactic variables. A subsequent encoding of initial strings is a representation of the rewrite rules. A simple tally scheme computes the experimental frequency-defined probabilities.

The model described above is a blueprint for a language discovery machine. Such a machine has been implemented in <u>A Programming Language</u> (APL): this machine has been tested on a fragment English grammar and on several formal grammars. The fragment English grammar consists of 87 rules on 52 words in 23 classes; the rules govern the 18 syntactic variables in this syntax-controlled probability grammar. The teacher-learner interaction is portrayed with no explicit semantics and no environment (context). That is, semantics and pragmatism are contained in neither the teacher or learner nor the training sequence. The language strings appear to have a semantic aspect: this is built into the syntactic rewrite rules. The expected sentence length (computed from the mathematical model) is 7.05 words. In a typical experiment, 115 sentences were heard by the learner to determine 20 of the 23 classes and to correctly classify 48 of the words in the dictionary. After 19 sentences were analyzed, all of the 18 variables were discovered. The graph in figure 4.1 illustrates the learning characteristics exemplified by the word class discovery procedure. More complete experimental results and a mathematical model for the word class learning appear in later sections.

Preliminaries

The following sections, which present no new results, contain the definitions and results from formal language theory and automaton theory which serve as research background for syntactic abduction of linear strings presented in the later chapters. The exposition follows Hopcroft and Ullman [1969]. The exposition of the syntax-controlled probabilities follows Grenander [1967].

Phrase Structure Languages

1. Let V_T denote a finite set of symbols called terminals or words; let V_T^+ denote the set of all finite length strings of these words; and let V_T^* denote $V_T^+ \cup \{NULL\}$ where the empty string of length zero is denoted by NULL. A language is any element of the powerset of V_T^*. Those (possibly infinite) subsets of V_T^* which have finite generating representations are called recursively enumerable. These finite generating representations or specifications are called phrase structure grammars and are formulated as follows: introduce an auxiliary finite set of symbols, denoted by V_N, called non-terminals or syntactic variables, with a distinguished symbol $S(\in V_N)$; introduce a finite set of rewrite rules R which govern these variables and which is a subset of $V_N^+ \times V^*$, where V denotes $V_N \cup V_T$. Then the phrase structure language consists of the strings which can be derived from S (the start symbol) by successive application of the rules. This is rigorously described by the introduction of a relation from V^+ to V^* as follows: for any $u \in V^+$ and $v \in V^*$, u is said to directly derive v (in the grammar) if there are strings $i,j,x,y \in V^*$ such that $u = xiy$, $v = xjy$ and $(i,j) \in R$. This can be extended by saying that u derives v in the grammar if either $u=v$ or if there is a finite sequence $Z_o, Z_1, Z_2, \ldots, Z_m \subset V^*$, $m > 1$, such that $u=Z_o$, $v=Z_m$, and Z_i directly derives Z_{i+1} for $i=0(1)m-1$. Then the language generated by this grammar is defined as the set of strings of words which can be derived from the start symbol S.

If S derives u (denoted by $S \rightarrow u$) and u contains variables, then u is called a sentential form. Note that the elements in the set of rewrite rules, for example denoted by (i,j), are customarily also written as $i \rightarrow j$. If two grammars generate the same language, then they are said to be weakly equivalent.

2. Context-Free Languages. If the rewrite rules R are restricted to finite subsets of $V_N \times V^*$, then the resultant grammar is called a context-free grammar.

3. Finite-State Languages. Those subsets of context-free grammars to which we focus our attention are called finite state grammars. The variables are governed by rewrite rules of two types: continuing $i \rightarrow xj$ or terminating $i \rightarrow x$, where $i,j \in V_N$ and $x \in V_T$. Denote the finite set of rules which rewrite i by R_i and assume that the generating algorithm begins by the application of a rule selected from R_1.

The language generated by such a grammar, which consists of V_T, V_N and R, is denoted by L(G); the symbol G denotes the triple (V_T, V_N, R) and R denotes the finite set of all rules. The language L(G) consists of all those strings in V_T^* which can be produced by the application of the rules in R.

4. Relation to Finite State Automata. The language generated by a grammar is some set of strings as described above. This set is also the set accepted by some finite state automaton.

The identification of $L(G)$ with its finite state automaton acceptor proceeds as follows: the variables in V_N, of which there are n_v, correspond to the states $1,\ldots,n_v$ where the state 1 corresponds to the distinguished variable S introduced above and, in addition, we introduce a final state F which is the "target" state for those variables which are governed by terminating rewrite rules.

5. Syntax-Controlled Probabilities. For each $R_i = \{r_{j_1}, r_{j_2}, \ldots, r_{j_{n_i}}\}$, where $r_{..}$ denotes a rule and n_i is the number of rules rewriting i introduce a probability distribution over R_i so that

$$\sum_{k=1}^{n_i} P(r_{j_k}) = 1,$$

where $i=1,2,\ldots,n_v$.

6. Markov Chain. Consider the application of the grammar G together with the probability distribution. In terms of the automaton description, the probability that the machine will be at state ℓ at time $t+1$ given that it is at state k at time t is specified. A system which evolves through a finite number of states (n_v+1) with a specified conditional probability of transition between two states for a given state at time t which is independent of t is called a finite homogeneous Markov chain (Kemeny and Snell [1960]). The familiar state diagram for finite automata (with labeled arcs which indicate letters in V_T and the probability) has a description in terms of two matrices: a matrix of probabilities and a matrix which prescribes the letters which correspond to transitions between states.

Chapter 2

The Word Class Discovery Algorithm

1. Problem Statement

The grammar G defined on the word dictionary V_T induces an equivalence relation on V_T in a natural way. This equivalence relation is based on grammatical substitutability. Introduce a Boolean function g, the grammaticality function defined on the set of all strings V_T^* so that

$$g(u) = \begin{cases} 1 & \text{if } u \text{ in } L(G) \\ 0 & \text{otherwise.} \end{cases}$$

Then the word equivalence relation (EQUIV) induced on V_T is precisely defined by:

for x,y in the finite set V_T, x EQUIV y if

g(uxv) = g(uyv) for all u,v in V_T^*.

Define a partition of V_T to be a set of disjoint non-empty subsets of V_T whose union is V_T. We have the following

Lemma. The relation EQUIV on V_T induces a partition of V_T into classes CL[J], J=1,2,...,NC. For each J, and any I≠J

1. x,y in CL[J] implies TRUE = x EQUIV y and
2. x in CL[I], y in CL[J] implies FALSE = x EQUIV y.

Denote the empty set by NULL. Denote the set difference of A and B by A CMPL B (also called the complement of B in A). Then the proof proceeds as follows:

```
[1]   NC ← 0
[2]   A ← W
[3]   L1: → L2  IF  A = NULL
[4]   NC ← NC+1
[5]   Pick x∈A
[6]   CL[NC] ← {y∈A:y EQUIV x}
[7]   A ← A CMPL CL[NC]
[8]   → L1
[9]   L2: → L3  IF  NC≠0
[10]  NC ← 1
[11]  →  L2
[12]  L3: → 0
```

The problem is to carry out step [6]. The formation of the set $\{y \in A$: y EQUIV x$\}$ must be carried out over V_T^* which is (denumerably) infinite.

2. We now develop the algorithm to approximate the partition which is induced by the infinitary equivalence relation EQUIV. Make the following observation: for words x and y, if g(uxv)≠g(uyv) for any strings u and v, then this is sufficient to conclude that words x and y are not equivalent. However, for words x and y and for some particular strings u and v, it is possible that g(uxv) = g(uyv) but x and y are not equivalent. It is for this reason that the usual procedures for partitioning a set under an equivalence relation (Knuth [1973]) are not recommended. Procedures based on aggregation of classes (or coalescing, or establishing links between classes) might

erroneously combine two classes on the basis of a single positive result of the test for simultaneous grammaticality of strings containing the two tested words. The algorithm detailed in the next section is based on the decomposition of the dictionary into classes; the classes are formed as they are needed.

3. An Approximating Algorithm. Here is a learning procedure which provides an approximating algorithm for the partition of the dictionary into word equivalence classes. We assume that the entire dictionary is known; the dictionary is unfolded into the classes as described below.

With the observation noted in the previous section, suppose that x and y are at some stage classed together and that for strings u,v, $g(uxv) \neq g(uyv)$; then there is reason to reclassify one of the words. The algorithm begins with the dictionary as the only equivalence class; new classes are formed as the need to create them occurs. Each class is characterized by a unique, <u>fixed</u> representor word called a <u>prototype</u>. Once a class has been established, there is no mechanism to remove it from the partition.

A. A word chosen from the dictionary is used as the prototype of the (only) word class. This is the tabula rasa hypothesis about the dictionary: that the partition consists of one class with the chosen word as the class prototype, once chosen, this word is fixed for all time as this class prototype. After this initialization is carried out, the algorithm proceeds according to the graphical descriptions given below. In these and in subsequent chapters, the algorithms will be described in several ways among which are: flow diagrams, block contour model diagrams with APL used as a medium for the sequential, procedural, high-level language, and English language descriptions. Note that these algorithms have been implemented on a digital computer via an IBM VS APL processor; the dependence upon "system environment" and elaborate input-output protocols have been minimized.

LISTEN produces a string in L(G). ATTENTION generates a pointer NUM to the string. The mechanism to generate NUM will be described later.

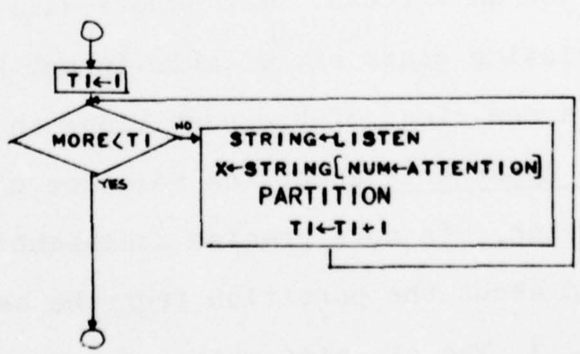


Figure 2.1

B. The PARTITION of V_T proceeds as follows:

1. Suppose that the word X is not the prototype of an existing class of the partition. Then we must ESTABLISH BELIEF about x: either x is to be classified in an existing class or, if this is not possible, then x forms a new class of the partition.

Note that if the class membership of every word were determined, then the partition would be known; that is, if we ESTABLISH BELIEF about a word x relative to the partition, then this ESTABLISHes BELIEF about the partition itself.

ESTABLISHing BELIEF about a word. The word x is classified to an existing class CL[k] if the string generated by LISTEN is grammatical with the prototype of CL[k] substituted for x in this string. The prototypes of the existing classes are the elements of PROTOS. These are tested successively (relative to the current string) beginning with the prototype of the current class to which x belongs. If the current classification of the word x is not correct, then the current partition will be modified as follows. The word x will either be reclassified to an existing class or, if this is not possible, the word x will form a new class of the partition; these two cases introduce <u>a new hypothesis</u> about the relation of the word x to the current partition. (A more precise statement: the new hypothesis is formed about the partition from the set of all partitions of the set V_T.) The classification of a word x is called ADJUST; the formation of a class is called ADD. These are detailed in the sections which follows.

Modifying the partition: ADJUST and ADD. In a particular implementation of the algorithm, the dictionary of words is a linear list. The first element of this list is the prototype of the initial <u>tabula rasa</u> word class. A second list, called PROTOS, will record the prototypes.

As noted earlier, if a word x cannot be classified to an existing class (relative to the current string), then a new class is added to the partition. The word x is to be the prototype of this new class. The word x is moved to the end of the list and PROTOS is updated. As classes are formed, the list of words will be subdivided into sublists which correspond to the discovered classes of the partition. Each sublist is initially defined by a prototype.

Suppose that at some stage a word x is to be reclassified. Then it is moved from its current position in the list to the end of the sublist of the class to which x is believed to belong. If x is reclassified to the last class formed, then x is deleted from the list and pushed onto the end of the list; (this move will be called LAST). Otherwise, x is deleted from the list and inserted into the list (by a push onto the sublist which constitutes the correct class, relative to the current string). The insertion move is called PROMOTE.

Suppose that relative to the current string, x is tested equivalent to the prototype of its current class. Then within this (linear) sublist, the word x is moved closer to its prototype; if x is already adjacent to the prototype, no action is taken. This type of move is called STRENGTHEN.

Note that in the case that a word is removed from a class it cannot return to this class. If a word x is to be removed from a class whose prototype is p, this means that for the current STRING rewritten as uxv we have

$$g(uxv) \neq g(upv).$$

That is, the STRING which separates x and p has occurred; there cannot be any reason to retry the class to which p belongs. Suppose the classes are assigned indices beginning with one which corresponds to the initial <u>tabula rasa</u> class. We restate the above by: once x has been promoted to a higher index class, the class index need never be lowered. In some sense, words are bubbled through sublists unidirectionally toward the correct classes. Within a sublist, the movement is toward the class prototype.

In summary, to ESTABLISH BELIEF about the partition for a non-prototype word x relative to its occurrence in a string we either ADD (if the existing classes are rejected as candidates for the class to which x might belong) or ADJUST. In the case of ADJUST, we either <u>strengthen</u> or move the word x via PROMOTE/LAST.

2. Suppose that the selected word x is a prototype; then it defines what is believed to be a class of the partition. In this case, a REFINEment of the believed partition <u>might</u> be possible. The procedure described here is made necessary by the fact that the prototype is fixed for a class which may, for example, contain words incorrectly classified. In particular,

suppose that for such an incorrectly classified word y the following is true:

every grammatical sentence which contains y is grammatical with x substituted for y but not conversely. This is of course possible and corresponds to a grammar which contains the following rules:

Rules

$$V_1 \rightarrow xV_2$$
$$V_1 \rightarrow yV_3$$
$$V_2 \rightarrow aV_4$$
$$V_3 \rightarrow aV_4$$
$$V_3 \rightarrow bV_5$$
$$V_4 \rightarrow bV_5$$

(or an automaton with the following segment):

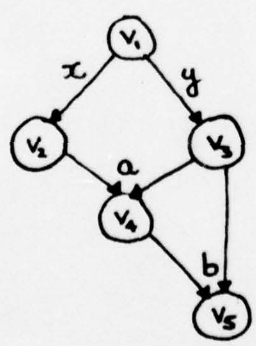


Figure 2.2

Suppose that x is the prototype of the word class to which y is believed to belong. Then since every string $L(G)$ which contains y is grammatical with x substituted for y, the fact that they are not equivalent will not be discovered by the procedure ESTABLISH BELIEF. This would lead to the erroneous conclusion for long times.

To allow for this possible REFINEment of the partition, we proceed as follows: if the word x is presented for classification and x is a class prototype, then a word distinct from x in this class (if any) is selected for the grammaticality test in the current string. If this word y is found to be not equivalent to x relative to the current string, then this is sufficient to guarantee that x is not equivalent to y, that y cannot belong to this class and so we abduce the new hypothesis:

that y belongs to a higher index class. We act on this hypothesis by a PROMOTE, LAST or ADD as appropriate.

If this word y is found equivalent to x, then by dint of the grammaticality of the current sentence with the word y substituted for x the strength of belief in the word y relative to the partition is increased; that is, the word y is moved toward the prototype x unless it is already adjacent to x in the linear list.

C. 1. For every STRING generated by LISTEN, the ATTENTION function measures the "strength of belief" of the words in STRING relative to the current partition by the distance of each word in STRING from the prototype of its current class. Then the selection of a word in STRING to be classified to the partition is based on the random selection of the candidate words weighted by this distance. Recall that, as the algorithm is carried out, the strength of belief of a non-prototype word which is successfully (i.e., in the affirmative) tested equivalent to its believed prototype is increased by actually moving the word (or its pointer) toward the prototype in the sublist which contains it. This is a concise representation of the partition at any time which has a "built in" relative measure of the strength of belief of the words to the sought-for partition.

2. In the case that the word x selected by ATTENTION is a prototype, the selection of the word y within this class (if this class contains at least two words) is carried out in an analogous way. That is, the selection favors words whose strength of belief is small.

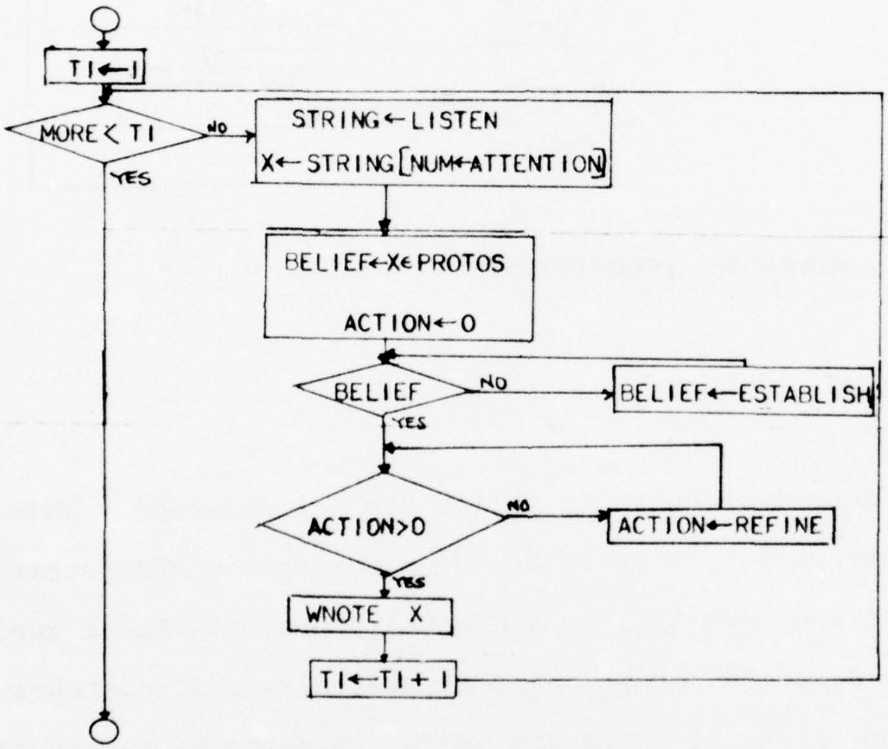
3. In an early experiment with this algorithm (and the model based on the data described in detail in the next chapter) the ATTENTION function selected the word x to be classified by a

simple randomization scheme. That is, each word in STRING (counting any word only once) is a candidate for classification to the partition. This seemed to carry out the partition discovery at a slower rate than the improved "strength measuring" scheme.

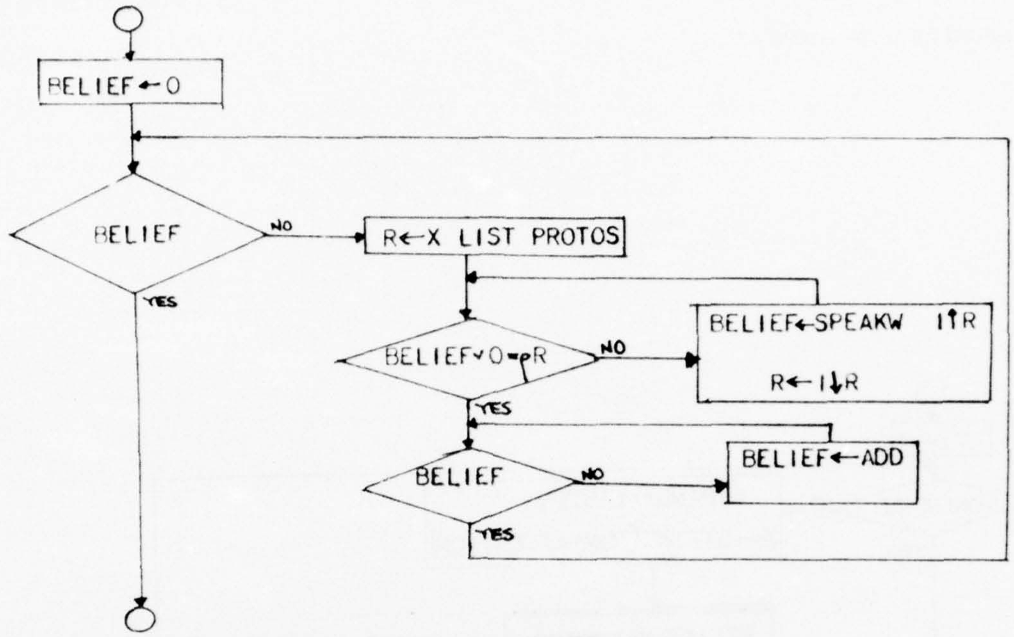
We note also that the weighted selection scheme described above is made efficient by a device which is only approximately proportional to strength of belief. The degree of the approximation improves for large numbers of elements.

D. The early experiments did not include the "sub-algorithm" REFINE and so, in principle, they could not work for all finite state grammars. Even with the REFINE algorithm as implemented, which obviously converges after some sufficiently long time, the convergence might be very slow. This will be made clearer in the next chapter with reference to particular examples. We note here that this slow convergence is due to the possibility that a word selected and removed (PROMOTED) from a large class on the basis of REFINEment might be promoted to a small class! That is, the PROMOTE might move the word to the next higher index class on the basis of rejection from the large, current class. However, an inordinately long time, from the practical point of view might pass before an "improved" classification may take place. This could be avoided in a manner similar to that to be used in state discovery discussed later. It will require some additional overhead in terms of storage of information, but it will reduce the discovery time significantly. In future work, this will be done; that is, words PROMOTEd by the REFINEment will be "tagged" to

force selection for classification when a sentence which contains the word(s) occurs.



ALGORITHM: PARTITION INTO WORD CLASSES



ALGORITHM: ESTABLISH BELIEF

Figure 2.4

4. The descriptions which follow use APL notation. This notation has proved useful to describe the algorithms and to subsequently implement and test them on a digital computer. The algorithms are also described graphically by nested sets of contours (Johnston [1971]); these are useful to describe the dynamic processes defined for computer structures. Some of the support functions and the details of the data structures appear in appendix A, pp. 121-125.

The monadic function PW MORE processes MORE sentences in an environment in which PW sentences have already been processed. The usual graphical topology for PW MORE, i.e. a flow chart, was drawn in section 3 above.

BIRTH randomizes the word list and initialize the sentence counter TW.

```
∇ BIRTH
PROTOS←,(C←(C←NV+⍳NW)[NW?NW])[1+TW←0]
```

Figure 2.5

PW processes MORE sentences. It invokes LISTEN which produces the string to be processed

```
 ∇ PW MORE;T1;NUM;STRING;X;BELIEF;ACTION
 T1←1
NEWS:→OUT IF MORE<T1
   STRING←LISTEN
   X←STRING[NUM←ATTENTION]
   BELIEF←X∊PROTOS
   ACTION←0
  L1:→L2 IF BELIEF
     BELIEF←ESTABLISH
     →L1
  L2:→NEXT IF ACTION>0
     ACTION←REFINE
     →L2
  NEXT:WNOTE X
   T1←T1+1
   →NEWS
OUT:→0
```

Figure 2.6

LISTEN invokes REWRITE which generates a sentence in the language; this string and the updated counter TW is printed.

```
∇ STRING←LISTEN;B;NB
(B,0ρNB←ρB←CR,(⍕TW←TW+1), '. ' ),PRINT STRING←REWRITE
```

Figure 2.7

ATTENTION produces a pointer to the word in the current sentence which is selected for classification

```
∇ NUM←ATTENTION;INDICES;S;P;CD;L;DP;RESULT
INDICES←C⍳PROTOS
CD←+⌿INDICES∘.≤P←C⍳S←¯1↓STRING
DP←P-INDICES[CD]
NUM←RESULT[?⍴RESULT←(RESULT←DP∊⌈/DP[?2⍴L])/⍳L←⍴S]
```

Figure 2.8

ESTABLISH carries out classification as described in the text in earlier sections.

Figure 2.9

```
 ∇ BELIEF←ESTABLISH;R
 BELIEF←0
L1:→OUT IF BELIEF
   R←X LIST PROTOS
  L2:→L4 IF BELIEF∨0=⍴R
      BELIEF←SPEAKW 1↑R
      R←1↓R
      →L2
  L4:→L1 IF BELIEF
      BELIEF←ADD
      →L4
OUT:→0
```

For a list of word class prototypes P and word x, LIST returns that sublist of P beginning with the class to which x is believed to belong.

Figure 2.10

```
∇ Z←X LIST P;IX
IX←C⍳X
Z←(¯1++/IX≥C⍳P)↓P
```

SPEAKW: for word RJ substituted in the sentence: if grammatical, invoke ADJUST and return BELIEF true; otherwise, BELIEF returned false

```
  ∇ BELIEF←SPEAKW RJ;RETURN
  BELIEF←0
  RETURN←~ACCEPT(STRING[⍳NUM-1],RJ,STRING[NUM+⍳(⍴STRING)-NUM])
L1:→OUT IF RETURN
     BELIEF←ADJUST RJ
     RETURN←1
     →L1
OUT:→0
```

ADJUST carries out modification to the partition (strengthen, promote, last) as appropriate.

```
∇ BELIEF←ADJUST RJ;IX;R1;R
 BELIEF←0
 IX←C⍳X
 ACTION←2
L1:→L3 IF BELIEF∨~RJ=1↑R←X LIST PROTOS
   BELIEF←(C⍳RJ)=¯1+IX
  L2:→L1 IF BELIEF
     BELIEF←STRENGTHEN
     →L2
L3:→OUT IF BELIEF
  L4:→L6 IF BELIEF∨RJ=¯1↑R
     R1←¯1+C⍳R[1+R⍳RJ]
     BELIEF←PROMOTE
     →L4
  L6:→L3 IF BELIEF
     BELIEF←LAST
     →L6
OUT:→0
```

Figure 2.12

Moves X at IX closer to its prototype.

```
∇ Z←STRENGTHEN;I
 C[I]←C[⌽I←(¯1+IX),IX]
 ACTION←3
 Z←1
```

Figure 2.13

Moves X to target class RJ (rightward)

```
∇ Z←PROMOTE
 C←C[(⍳¯1+IX),(IX F R1),IX,R1 F⍴C]
 ACTION←4
 Z←1
```

Figure 2.14

Move the element X at IX in the list C to the end of the list (rightward).

```
∇ Z←LAST
 C←C[(⍳¯1+IX),(IX F⍴C),IX]
 ACTION←5
 Z←1
```

Figure 2.15

F: Cuts a list of B elements at A and returns pointers to elements at the right of A.

```
∇ Z←A F B
Z←A+⍳B-A
```

Figure 2.16

Add the word X to the partition as a prototype.

```
∇ Z←ADD;IX
PROTOS←PROTOS,X
IX←C⍳X
Z←LAST
ACTION←6
```

Figure 2.17

REFINE is invoked if the current word is a class prototype; this algorithm selects a word in the class for testing.

```
 ∇ ACTION←REFINE;SZ;R;BELIEF;FLAG
 ACTION←1
 SZ←⍳¯1+(-R-(1↓R←C⍳PROTOS),1+⍴C)[PROTOS⍳X]
 BELIEF←0=⍴SZ
L1:→OUT IF BELIEF
   X←C[(C⍳X)+⌈/SZ[?2⍴⍴SZ]]
   BELIEF←0
   R←X LIST PROTOS
   FLAG←ACCEPT STRING[⍳NUM-1],X,STRING[NUM+⍳(⍴STRING)-NUM]
  L2:→L3 IF FLAG
     R←1↓R
     FLAG←1
     →L2
  L3:→L4 IF BELIEF∨0=⍴R
     BELIEF←ADJUST 1↑R
     →L3
  L4:→L1 IF BELIEF
     BELIEF←ADD
     →L4
OUT:→0
```

Figure 2.18

WNOTE reports on the actions taken.

```
 ∇ WNOTE X;FLAG;A
 A←(7ρ' '), 'STRING[' ,(⍕NUM),']'
 ⍞←A,(PRINT,STRING[NUM]),' ', 'ω¨←↑L+↓' [ACTION]
 FLAG←X=STRING[NUM]
L1:→OUT IF FLAG
 ⍞←' ',PRINT X
 FLAG←1
 →L1
OUT:' '
```

Figure 2.19

Chapter 3

A Fragment of English Grammar

1. In these sections we describe a fragment of English grammar which serves to test the word discovery algorithm. This grammar consists of words and rewrite rules with a probability distribu- on the rules. The grammar can be driven to randomly generate English-like sentences in this syntax-controlled probability language. Some sample sentences are:

1. It is orange.
2. She dislikes the dog and a chair was not blue while John speaks.
3. It is not orange.
4. It was green and a dog was not seen by the girl.

2. The 52 words listed below are arranged in 23 labeled word class as shown in table 3.1:

LABEL	WORDS	Probability in class	Descriptor
DET	a	3/10	determiner
	some	1/2	
	the	1/5	
AJH	clever	1/4	human adjective
	short	1/4	
	tall	1/4	
	young	1/4	
AJA	frisky	1/2	animal adjective
	spotted	1/2	
AJ1N	fine	1/4	neuter adjective group 1
	new	1/4	
	valuable	1/2	
AJ2N	blue	1/3	neuter adjective group 2
	green	1/3	
	orange	1/3	
NH	boy	1/4	human noun
	girl	1/4	
	man	1/4	
	woman	1/4	
NA	cat	3/10	animal noun
	dog	1/5	
	kitten	3/10	
	puppy	1/5	
NN	chair	1/3	neuter noun
	desk	1/3	
	table	1/3	
AUX	is	1/2	auxiliary verb
	was	1/2	
VP	helped	1/3	passable verb
	hurt	1/3	
	seen	1/3	
VT	dislikes	1/2	transitive verb
	likes	1/2	

Table 3.1

LABEL	WORDS	Probability in class	Descriptor
VI	sings	1/2	intransitive verb
	speaks	1/2	
CONJ	and	4/5	conjunction
	while	1/5	
PRH	he	1/2	human pronoun
	she	1/2	
PRN	it	1	neuter pronoun
PNH	John	1/2	human pronoun
	Mary	1/2	
PNA	Rover	1/2	animal pronoun
	Touka	1/2	
ADV	immensely	1/2	adverb
	violently	1/2	
VC	claims	1/2	verb (claim)
	says	1/2	
REL	that	1	relative
BY	by	1	instrumental
NOT	not	1	negation
DOT	.	1	period

Table 3.1 - continued

3. The phrase structure formulas of the rewrite rules over this dictionary are:

Formulas	Comments
S → NP+AUX+VP1	neuter
NP → PRN|DET+(AJ2N)+NN	
VP1→ (NOT)+AJ2N	adjectival predicate
S → NP+AUX+VP1 NP → PNA|DET+(AJA)+NA VP1→ (NOT)+VP+BY+NP1 NP1 → DET+NA∪NH	animal
S → NP+AUX+VP2|NP+VP3 NP → PRH ∪PNH VP2 → (ADV)+VT+NP1|VI NP1 → DET+NA UNH VP3 → (NOT)+VP+BY+NP2 NP2 → DET+(AJH)+NH	human
S → NP+VC+THAT+S NP → PRH ∪PNH	relatives
S → S+CONJ+S	globals

Table 3.2

4. These rules are expressed in another form below. Introduce syntactic variables, denoted by 1,2,3,...,NV=18. The generating algorithm begins by the application of a rule which rewrites variable 1. The probability of each rule is shown in the last column.

Table 3.3. Fragment grammar rewrite rules

#	Rewrite Rule			Probability
1	1 →	DET	2	1/4
2	1 →	PRH∪PNH	6	1/4
3	1 →	PNA	7	1/4
4	1 →	PRN	8	1/4
5	2 →	AJH	3	1/6
6	2 →	AJA	4	1/6
7	2 →	NH	6	1/6
8	2 →	NA	7	1/6
9	2 →	AJ1N	5	1/6
10	2 →	NN	8	1/6
11	3 →	NH	6	1
12	4 →	NA	7	1
13	5 →	NN	8	1
14	6 →	VI	12	1/5
15	6 →	ADV	17	1/5
16	6 →	VT	9	1/5
17	6 →	AUX	13	1/5
18	6 →	VC	18	1/5
19	7 →	AUX	13	1
20	8 →	AUX	10	1
21	9 →	DET	11	1
22	10 →	AJ2N	12	1/2
23	10 →	NOT	16	1/2
24	11 →	NH∪NA	12	1
25	12 →	CONJ	1	1/5
26	12 →	DOT	0	4/5
27	13 →	VP	14	1/2
28	13 →	NOT	15	1/2
29	14 →	BY	9	1
30	15 →	VP	14	1
31	16 →	AJ2N	12	1
32	17 →	VT	9	1
33	18 →	REL	1	1

5. Each word class shown above has a probability distribution on it which governs the selection of a word for the application of the rewrite rule as shown in the word table in section 2. The syntax-controlled probability fragment English grammar in the finite state form is obtained by the corresponding expansions of the rules in section 4. When this expansion is carried out we get 87 rules. For example, the rewrite rules for variable 1 with the probabilities of application are

R_1	Probability
1 → a 2	3/40
1 → some 2	1/20
1 → the 2	1/8
1 → he 6	1/8
1 → she 6	1/8
1 → Mary 6	1/8
1 → John 6	1/8
1 → Rover 7	1/8
1 → Touka 7	1/8
1 → it 8	1/4

6. The graph of the corresponding finite state automaton is shown below. For clarity, the arcs between the nodes are labeled with word class names. The target state for the terminating rules is denoted by F.

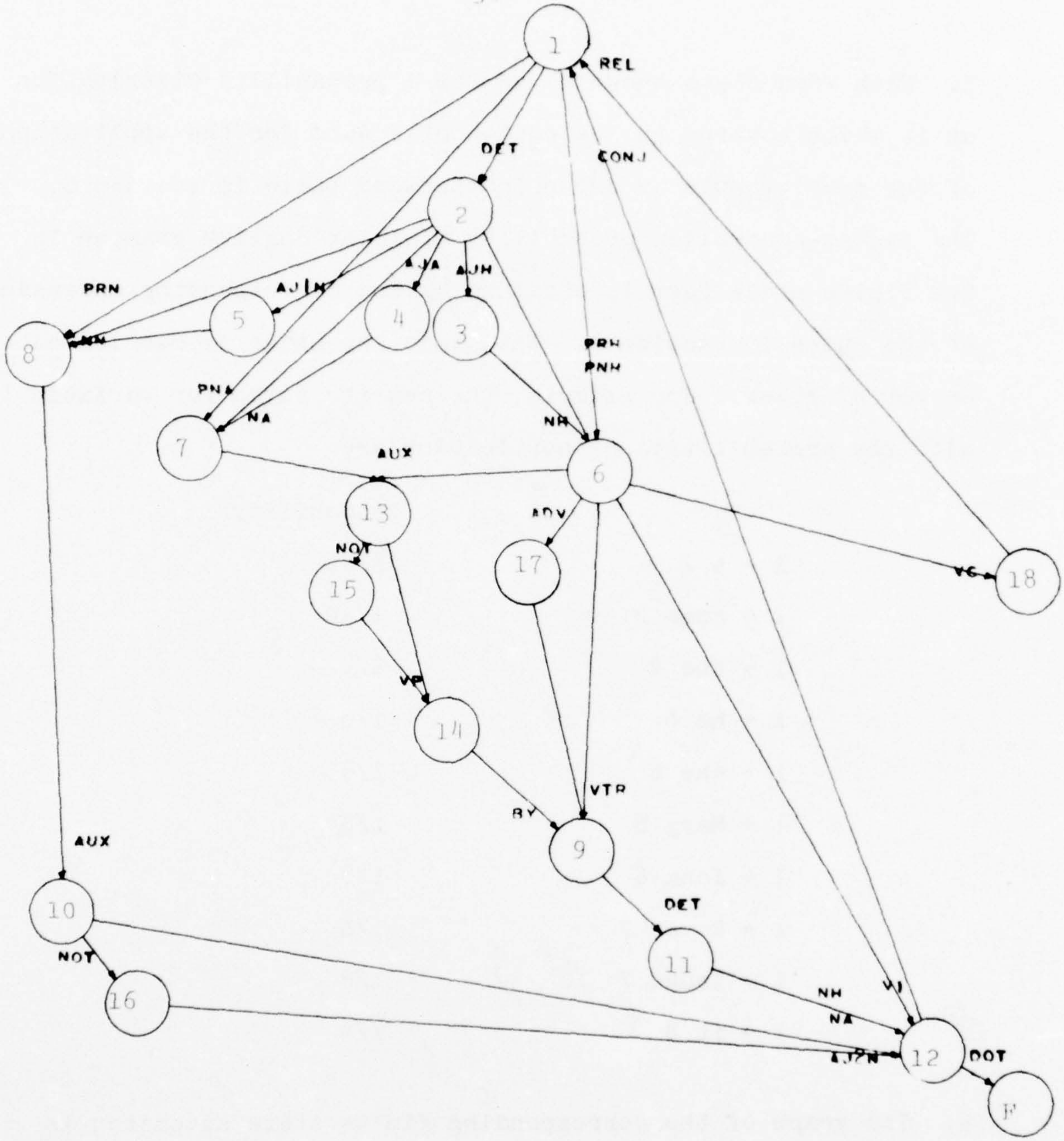


Figure 3.1. Fragment grammar transition diagram. This is the graph of the corresponding finite state automaton.

7. The syntax-controlled probability language L(G) is represented in data structures as described in Grenander (1974); the APL functions REWRITE (which produces a sentence in this probabilistic language L(G)) and ACCEPT (which returns the truth value of $u \in L(G)$ for any $u \in V^*$) are useful representations of L in the case that the deterministic form of the grammar is given. This latter remark is of course no restriction on the use of these procedures since the construction of the minimal state deterministic completely specified automaton (and its isomorphic language counterpart) is effective. A preprocessor can be implemented to prepare the data structures MATRIX and RULES. The details of the operation of these functions is left for another section.

The support function PRINT produces the "external" form of the words in the language L(G) which are internally represented most conveniently as integers. That is, the variables are 1,2,..., NV whereas the words are NV+1,NV+2,NV+3,...,NV+NW.

The modular APL functions carry out the abduction process which they formally describe. These functions can be easily modified to create variations of the algorithm and to assist in the design of extensions of this work. In particular, in addition to several variations, tests of robustness of word class discovery, a test for behavior to compare to a mathematical model, and the like were carried out with this abduction machine. An implementation in some more "production" oriented language is called for in the case of application of these techniques.

An experiment has been designed which enables a participant to act as the generator and the acceptor of strings in some

language. That is, the functions REWRITE and ACCEPT require "user" responses which enter the APL environment to carry out the abduction process. The results of these experiments will be reported in later work.

8. The detail of the experiment which follows is typical (modulo the random element) of the results of the experiments performed with this grammar for the abduction machine. Some directions for statistical analysis are presented in Chapter four; a more complete analysis will be carried out in later work when the computer implementations are oriented for the production of statistics in a large scale, cost-effective manner.

The word order within the dictionary is initially randomized. In the following experiment, the random number generator seed is preset to a given value (7*5). This is useful control to exert over probabilistic simulations; that is to say, it enables a reproducibility of results. This is essential to make the procedures "portable". This parameter setting is <u>not</u> essential to the interpretation of the results obtained. As can be seen from the word list below (compare section 2) the order of the words is scrambled by the function BIRTH. The fixed word class prototype of the <u>tabula rasa</u> word class relative to this scrambled order is "violently".

The function PW MORE is invoked with a value of more viz., how many MORE sentences are to be processed. The sentence generated by REWRITE is PRINTed, followed by the word selected by ATTENTION (STRING[NUM]). The special symbols indicate the classification of the word X←STRING[NUM]. In the case that X is

∈ PROTOtypes, the word tested in REFINE is also printed after the special symbol. The symbols are to be interpreted as follows:

+ ADD a class to the partition
↑ PROMOTE
← STRENGTHEN
¨ cannot STRENGTHEN: word is adjacent to the <u>fixed</u> prototype
ω selected word is PROTOS; word tested by REFINE is believed equivalent.
L word moved to last class.

9. In this experiment we note an early, *rapid word class* discovery with the time between discovery of classes increasing; this same "learning curve" characteristic is observed with respect to the number of words correctly classified. In the table below we summarize the establishment of classes with the word class PROTOtypes indicated.

TIME (sentence no.)	CLASS NO.	PROTOS
1	2	violently, sings
3	3	seen
4	4	by
5	5	a
6	6	that
7	7	Mary
10	8	frisky
11	9	girl
12	10	table
14	11	was
17	12	it
20	13	not
21	14	short
23	15	new
25	16	likes
28	17	orange
70	18	and
89	19	says
238	20	cat
852	21	Rover

```
⎕RL←7★5
BIRTH
PW 15

1.  HE SINGS .
      STRING[2] SINGS +

2.  A CHAIR WAS NOT BLUE WHILE JOHN SPEAKS .
      STRING[8] SPEAKS L

3.  MARY SPEAKS AND TOUKA IS NOT SEEN BY THE DOG .
      STRING[7] SEEN +

4.  ROVER WAS NOT HELPED BY A WOMAN .
      STRING[5] BY +

5.  MARY SAYS THAT MARY WAS HELPED BY A BOY .
      STRING[8] A +

6.  JOHN CLAIMS THAT TOUKA WAS NOT HELPED BY A KITTEN .
      STRING[3] THAT +

7.  MARY LIKES THE PUPPY .
      STRING[1] MARY +

8.  ROVER IS NOT HURT BY THE BOY .
      STRING[6] THE ↑

9.  TOUKA IS NOT HURT BY THE WOMAN .
      STRING[4] HURT ↑

10.  THE FRISKY DOG IS NOT HURT BY A GIRL .
      STRING[2] FRISKY +

11.  MARY DISLIKES SOME GIRL .
      STRING[4] GIRL +

12.  A FINE TABLE IS NOT ORANGE .
      STRING[3] TABLE +

13.  ROVER IS NOT HELPED BY SOME GIRL .
      STRING[6] SOME ↑

14.  JOHN CLAIMS THAT JOHN SAYS THAT ROVER WAS NOT HELPED BY THE CAT .
      STRING[8] WAS +

15.  JOHN VIOLENTLY DISLIKES THE DOG .
      STRING[1] JOHN ↑
```

PW 15

16. TOUKA IS SEEN BY THE KITTEN .
    STRING[6] KITTEN ↑

17. IT WAS BLUE .
    STRING[1] IT +

18. SHE WAS NOT HURT BY SOME WOMAN .
    STRING[7] WOMAN ↑

19. TOUKA IS SEEN BY THE WOMAN .
    STRING[5] THE

20. IT WAS NOT BLUE .
    STRING[3] NOT +

21. SOME SHORT BOY CLAIMS THAT MARY SPEAKS .
    STRING[2] SHORT +

22. MARY SPEAKS .
    STRING[1] MARY JOHN

23. THE NEW CHAIR IS NOT BLUE WHILE HE SPEAKS .
    STRING[2] NEW +

24. IT WAS NOT GREEN .
    STRING[1] IT ω

25. SHE IMMENSELY LIKES THE CAT .
    STRING[3] LIKES +

26. THE YOUNG BOY SAYS THAT TOUKA WAS HURT BY THE WOMAN .
    STRING[6] TOUKA ↑

27. SHE SINGS .
    STRING[1] SHE ↑

28. THE FINE TABLE WAS NOT ORANGE .
    STRING[6] ORANGE +

29. IT IS GREEN .
    STRING[2] IS ↑

30. IT IS NOT ORANGE .
    STRING[1] IT ω

PW 15

31. HE IMMENSELY DISLIKES SOME WOMAN WHILE SOME TABLE WAS GREEN AND A YOUNG WOMAN VIOLENTLY LIKES THE DOG WHILE ROVER IS NOT SEEN BY THE MAN .
STRING[25] THE ¨

32. MARY LIKES THE CAT .
STRING[3] THE ¨

33. MARY IMMENSELY DISLIKES THE CAT .
STRING[5] CAT ↑

34. HE LIKES A CAT .
STRING[1] HE ↑

35. THE KITTEN IS SEEN BY THE CAT .
STRING[6] THE ¨

36. ROVER WAS NOT SEEN BY SOME CAT .
STRING[1] ROVER ↑

37. JOHN WAS NOT HELPED BY A MAN WHILE THE FINE CHAIR WAS BLUE .
STRING[13] BLUE L

38. IT IS ORANGE .
STRING[1] IT ω

39. IT WAS GREEN .
STRING[1] IT ω

40. THE BOY LIKES A CAT .
STRING[1] THE ¨

41. IT IS NOT BLUE .
STRING[4] BLUE ¨

42. HE SINGS .
STRING[1] HE ←

43. ROVER WAS SEEN BY A CAT .
STRING[2] WAS ¨ IS

44. THE KITTEN IS SEEN BY A PUPPY .
STRING[1] THE ¨

45. TOUKA WAS HELPED BY THE CAT .
STRING[3] HELPED ↑

.
.
.

71. *HE VIOLENTLY LIKES THE MAN .*
STRING[2] VIOLENTLY ↑ VALUABLE

.
.
.

102. *HE SINGS .*
STRING[2] SINGS ↑ VALUABLE

.
.
.

135. *THE VALUABLE CHAIR IS NOT ORANGE .*
STRING[2] VALUABLE ↑

.
.
.

WORDS[C-NV;]

- • VIOLENTLY
- PUPPY
- IMMENSELY
- GREEN
- .
- YOUNG
- WHILE
- AND
- DOG
- CLEVER
- CHAIR
- SAYS
- SPOTTED
- MAN
- DISLIKES
- TALL
- DESK
- FINE
- CLAIMS
- VALUABLE
- BOY
- • SINGS
- SPEAKS
- • SEEN
- HURT
- HELPED
- • BY
- • A
- THE
- SOME
- • THAT
- • MARY
- JOHN
- TOUKA
- HE
- SHE
- ROVER
- • FRISKY
- • GIRL
- KITTEN
- WOMAN
- CAT
- • TABLE
- • WAS
- IS
- • IT
- • NOT
- • SHORT
- • NEW
- • LIKES
- • ORANGE
- BLUE

Table 3.5. Partitioned vocabulary at t=45

(The dark circle at the left of a word signals the prototype)

In this experiment, the subalgorithm REFINE is invoked for the first time at sentence number 22; it affirms the belief that: relative to the sentence "Mary speaks.", the word "John" (which on the basis of prior sentences was classed equivalent to "Mary") tests equivalent to the prototype "Mary". Moreover, since the position of the word "John" is adjacent to "Mary" in the list of words, the current strength of belief cannot be strengthened further.

At the 71st sentence, the word "valuable" is ejected from the first class by REFINE and this word is moved to target class two; the move is made on the basis of the non-grammaticality of the word "valuable" in place of the word "violently" in the current sentence. That is, the sentence "He valuable likes the man." is not grammatical in the fragment grammar. The word "valuable" is moved to target class three at t=102; it is correctly classed with "new" at t=135.

It is possible for a word class to be discovered for a word which does not appear in any sentence up to some time t. (Clearly, if a word does not appear for any time t, then it is removed from the dictionary.) This is illustrated by the "migration" of the word "clever" toward its class: "clever" is ejected from the first class at time 1137, and subsequent promotions are made at sentence nos. 1515, 1629, 1657 and 1678.

In early experiments which did not include the subalgorithm REFINE, the word classes in this fragment grammar were discovered anyway. In particular, the classes NA and NH were determined; this was due to the "adjectives", that is, the classes AJA and AJH.

To see this, suppose that, having established as classes [frisky] and [girl] (at t=11) we have the sentence

The frisky cat is seen by a girl.

Select the word "cat" for classification; then

The frisky [girl] is seen by a girl.

is not grammatical. Therefore, [cat] forms a new class.

We note that, without REFINE and without adjectives, the partitioning process is dependent upon the order in which the classes are determined; that is, if the class [cat] is established before [girl], then the discovery process converges to the sought-for partition.

10. An experiment in "robustness" was carried out by the introduction of a non-systematic error in the teacher. That is, the acceptor was modified by (randomly) negating the response 10% of the time. In this way, a grammatical sentence is reported non-grammatical and a non-grammatical sentence is reported grammatical.

The subalgorithms LIST and ADJUST were also modified. The list of words is treated as a circular list; movement of words during classification occurs both rightward and leftward. A subalgorithm DEMOTE was introduced to move words left.

At any stage of the algorithm, if the word to be classified cannot be classed to an established class, then it forms a new class. The results of this experiment with a bidirectional flow of words is that "spurious" word classes begin to appear. There is no mechanism for the consolidation of word classes; every word will eventually form its own class.

```
∇Z←X LIST P;IX
IX←C⍳X
Z←(¯1++/IX≥C⍳P)⌽P
```

Figure 3.2. Modified LIST

```
   ∇ BELIEF←ADJUST RJ;J;IX;R1;R
   BELIEF←0
   IX←C⍳X
   ACTION←2
  L1:→L3 IF BELIEF∨(⍴R)<⍴PROTOS
    BELIEF←(C⍳RJ)=¯1+IX
   L2:→L1 IF BELIEF
    BELIEF←STRENGTHEN
    →L2
  L3:→OUT IF BELIEF
    L4:→L7 IF BELIEF∨(⍴R←C⍳PROTOS)<1+J←PROTOS⍳RJ
      R1←R[J+1]-1
     L5:→L6 IF BELIEF∨R[J]<IX
       BELIEF←PROMOTE
       →L5
     L6:→L4 IF BELIEF
       BELIEF←DEMOTE
       →L6
    L7:→L3 IF BELIEF
      BELIEF←LAST
      →L7
  OUT:→0
```

Figure 3.3. Modified ADJUST

```
∇ Z←DEMOTE
C←C[(⍳R1),IX,R1 F⍴C]
ACTION←7
Z←1
```

Figure 3.4. DEMOTE

Chapter 4

Experimental Results and Data Analysis

1. After t sentences have been generated, we have observations (r, z_r), for $r=1,2,3,\ldots,t$, where z_r might represent the total number of words correctly classified or the total number of discovered word classes after the rth sentence has been processed. These data are the result of a probabilistic process so that each experiment from "BIRTH" will have different characteristics; it is hoped that these differences might be slight. It has been suggested that statistical regression might be applied to the results of several experiments. Another procedure might be to average over the several experiments the datum observed at each r and then to apply the data analysis described below to this smoothed, preprocessed data. Neither of these has yet been done nor explored further at this time.

2. The graphical representation of the data in figure 4.1, which is depicted as continuous for convenience, visually suggests <u>learning curve</u> properties. The data plotted on a semilog scale

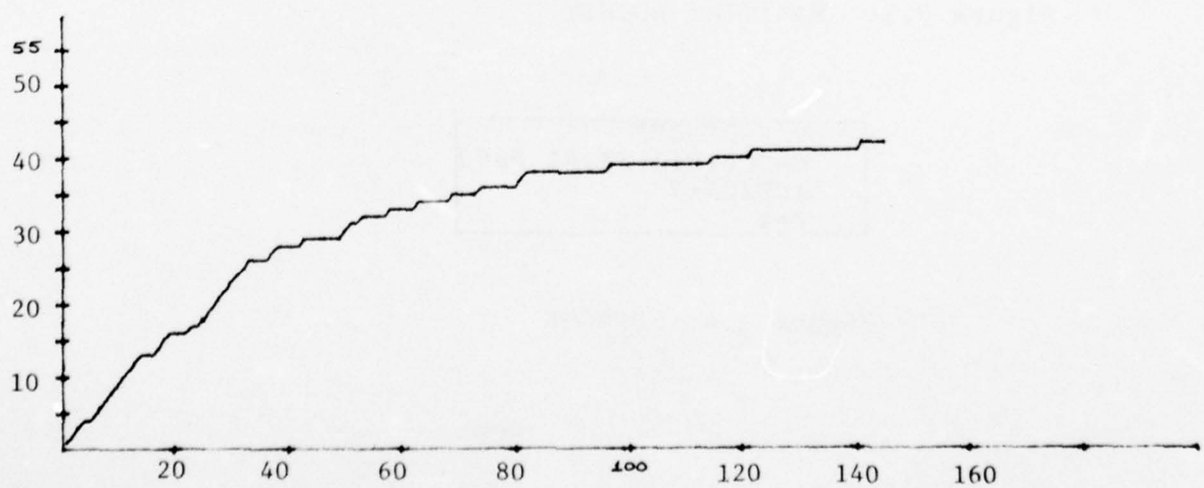


Figure 4.1. Number of words correctly classed v. number of sentences.

appears in figure 4.2. A least squares straight line fit to $(r, \ell n(z_a - z_r))$, where z_a denotes an asymptotic value, was determined and is superposed in figure 4.2 and was used to obtain the "fit" shown in figure 4.3 (Davis [1963]). The true asymptotic value, which in these experiments is of course known a priori, is increased by one so that in semilog coordinates (r,0) corresponds to "all words correctly classified."

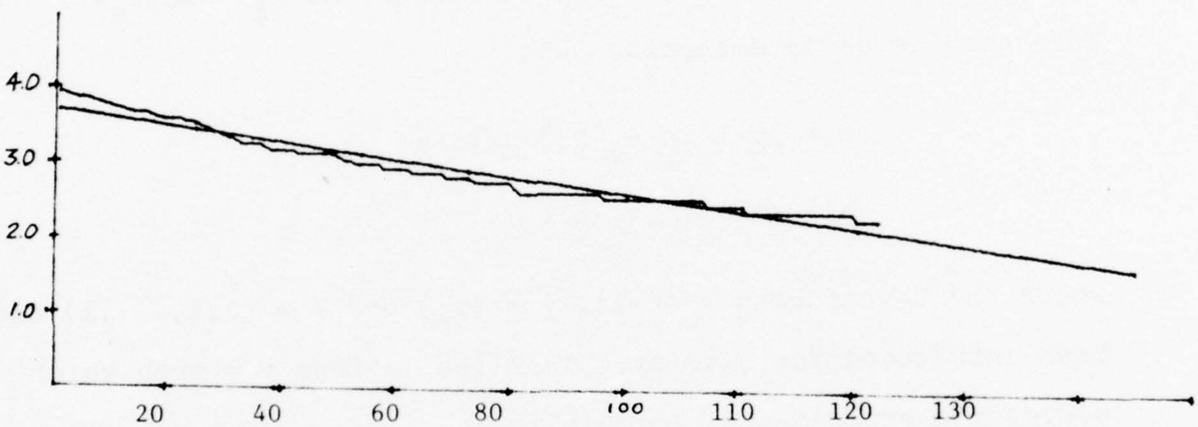


Figure 4.2: $\ell n(z_a - z_r)$ v. number of sentences and superposed fit as explained above.

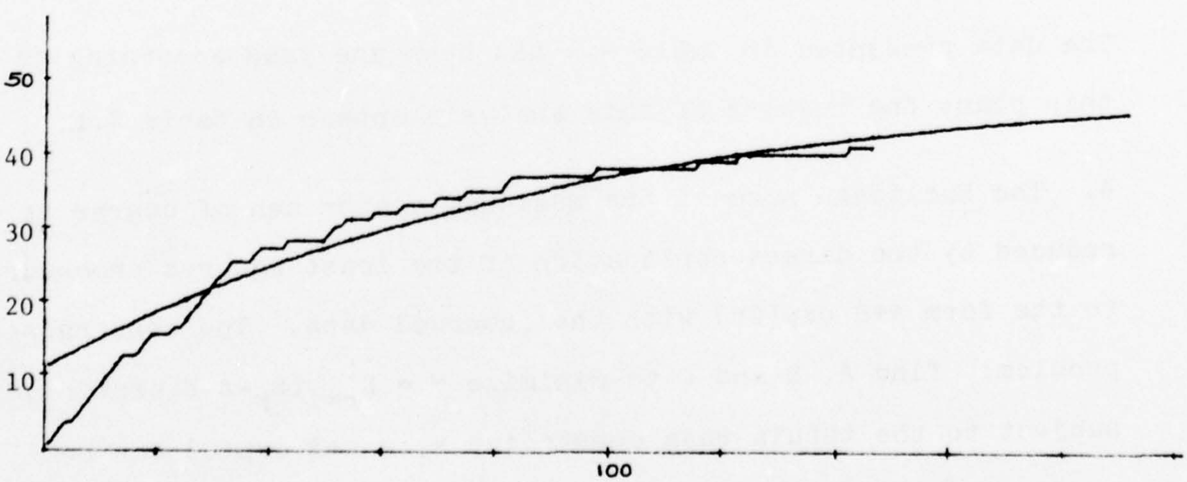


Figure 4.3: Number of words correctly classed v. number of sentences with superposed fit.

However, this fit does not have the important property which underlies the theoretical model: the tabula rasa hypothesis which makes natural a constrained least squares fit to force the curve through the point (1,1); this point corresponds to the initial equivalence class.

3. That is, letting $y_r = \ell n(z_a - z_r)$, we seek m and b to minimize $M = \Sigma_{r=1}^{t}(y_r - mr - b)^2$, subject to the constraint $y_1 = \ell n(z_a - 1) = m+b$. This enables us to determine that

$$m = [\underline{y} - \underline{1}\ \ell n(z_a - 1)]\cdot\underline{u}/(\underline{u}\cdot\underline{u})$$

$$b = -m + \ell n(z_a - 1),$$

where the t-vectors $\underline{u} = (r-1)$, $\underline{y} = (y_r)$ and $\underline{1} = (1,1,\ldots,1)$ have been introduced for clarity. The line is thus $y = mr+b$ which yields the experimental formula to be

$$z = z_a - \exp(mr+b) = z_a - (z_a - 1)\exp(-m)\exp(mr).$$

The data presented in table 4.1 has been analyzed according to this plan; the results of this analysis appear in table 4.1.

4. The Euclidean norm of the residual vector can of course be reduced by the direct application of the least squares procedure to the form A+B exp(Cr) with the observed data. The constrained problem: find A, B and C to minimize $M = \Sigma_{r=1}^{t}(z_r - A - B\exp(Cr))^2$ subject to the tabula rasa constraint $z_1 = A + B\exp(C)$ can be readily computed as follows:

The transcendental equation in C

$$(\underline{z}-1)\cdot(\underline{v}' - [(\underline{v}\cdot\underline{v}')/(\underline{v}\cdot\underline{v})]\underline{v}) = 0$$

where $\underline{v} = (\exp(C)-\exp(Cr))$ and $v' = \frac{dv}{dC}$ is solved for C and this value is used to determine $B = -\underline{v}\cdot(\underline{z}-\underline{1})/(\underline{v}\cdot\underline{v})$ and $A = 1-B\exp(C)$. This has been done and the results are also recorded in table 4.1 and depicted graphically in figure 4.4.

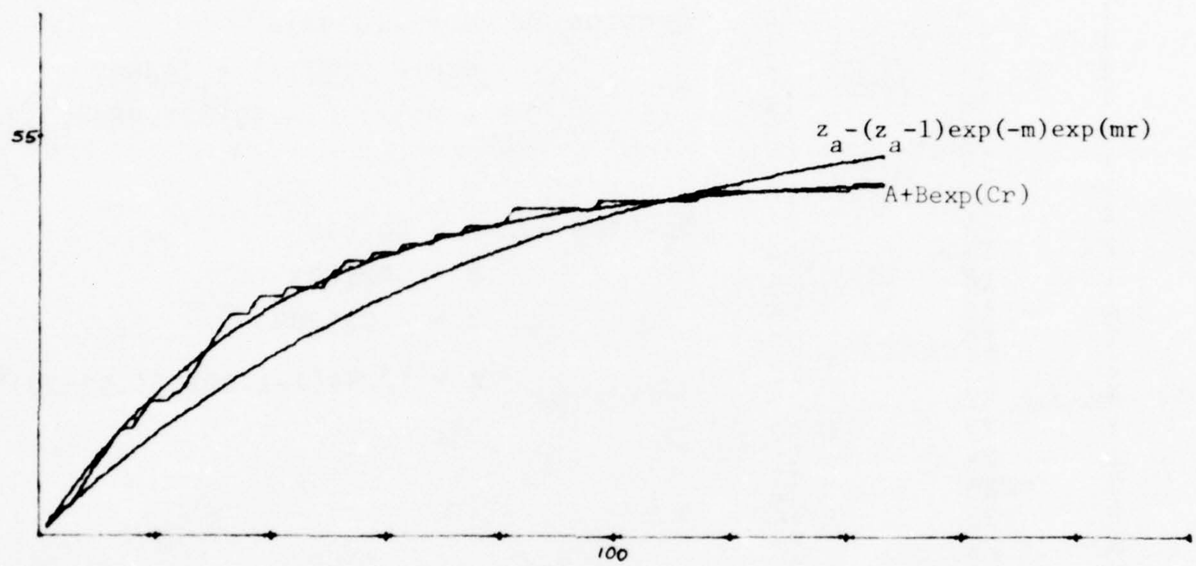


Figure 4.4: Three superposed graph as described in sections 3 and 4.

Δr	#words correctly classed	r
		0
1	1	1
1	2	2
1	3	3
2	4	5
1	5	6
1	6	7
1	7	8
1	8	9
1	9	10
1	10	11
1	11	12
1	12	13
3	13	16
1	14	17
1	15	18
4	16	22
2	17	24
1	18	25
1	19	26
1	20	27
1	21	28
1	22	29
1	23	30
1	24	31
1	25	32
4	26	36
1	27	37
5	28	42
7	29	49
1	30	50
2	31	52
5	32	57
5	33	62
6	34	68
5	35	73
7	36	80
1	37	81
15	38	96
18	39	114
7	40	121
19	41	140
5	42	145

Section 3: $m = -.013216$

$\exp(-.013216) = .98687$

$z = 52.\ [1-.99382(.98687)^r]$

Section 4: $A = 42.440$

$B = -42.493$

$C = -.025089$

$z = 42.44[1-1.00125(.97522)^r]$

TABLE 4.1

5. A conjecture based on visual observation is that the learning characteristics fall between the values determined in section 3 and section 4; the sought for characteristics would better represent both the early (rapid) learning and the asymptotic qualities viz., the time to determine all the words correctly. A heuristic procedure based on the observed data could be implemented; this would especially be of use if predictive qualities of the model were desired. In the least squares procedure, a weighting function appropriately chosen would improve the fit. We mention here that the true asymptotic value (e.g. the number of words to be classified), which is known a priori, denoted here by z_a could be introduced as an additional constraint to reformulate the least squares problem as: *find C to minimize* $\Sigma_{r=1}^{t}(z_r - z_a[1-(1-1/z_a)\exp(C(r-1))])^2$. This leads to a transcendental equation in C

$$[(\underline{z}-\underline{1}z_a)/(1-z_a) - \exp(C\underline{u})]\cdot\underline{u} = 0$$

where the t-vector $\underline{u} = (r-1)$.

Chapter 5

A Mathematical Model

1. Consider an "incidence matrix" of word equivalents whose entries are updated as the partitioning procedure is carried out. All the entries are initially set to some number p_1, $0 < p_1 < 1$. At each stage of the procedure, the entry corresponding to the pair of words selected for testing is either augmented or set to zero according as the test result for this pair is "believed equivalent" or "not equivalent" respectively; the other entries are unchanged. Thus the x,y entry in the matrix after the r+1st stage is

$$p_{xy}(r+1) = \begin{cases} p_{xy}(r) & \text{the pair } x,y \text{ not tested} \\ 0 & \textit{discovered that } x \not\equiv y \\ f(p_{xy}(r)) & \text{strengthened belief that } x \equiv y \end{cases}$$

where $f(\cdot)$ denotes *a function which augments* entries in the believed equivalent case.

2. The rate of convergence of this matrix to the <u>true</u> incidence matrix of the infinitary equivalence relation will be delayed by a non-zero probability of testing two words for equivalence and getting an "incorrect" result. That is, if two words x and y are not equivalent it is possible that for example, out of 100 sentences which involve the word x only 20 of these sentences would separate x from y; i.e. only 20 of these sentences would be ungrammatical with the word y substituted for the word x. This suggests that the ratio of the number of sentences which do not separate x from y to the total number of sentences involving x

(in the case that x and y are not equivalent) be considered as a candidate for the aforementioned probability; note that this quantity, which will be denoted by epsilon, depends on each pair of words. If epsilon is zero, then all grammatical sentences involving x separate x from y when they are not equivalent; in such a case, the non-equivalent words are separated as soon as they are presented together for testing. If epsilon is one, then every grammatical sentence involving x is also grammatical with y substituted for x and hence x.

3. The rate of convergence will also depend upon the frequency with which pairs of words are brought forth for comparison with respect to the equivalence relation. For fixed words x and y which are not equivalent, it is of interest to compute the rate at which the corresponding matrix entry p_{xy} converges to zero. Since the underlying process is probabilistic, we propose to compute the mean rate at which such an entry converges to zero. This will indicate how to estimate the mean time to determine the <u>true</u> incidence matrix and hence the mean time to determine all non-equivalent words. Let c_{xy} denote the probability that words x and y are brought forth for comparison. Then the expected value of the sum of the possible entries $p_{xy}(r)$ for non-equivalent words is estimated by

$$E[\sum_{x\neq y} p_{xy}(r)] = \sum_{x\neq y} E[p_{xy}(r)] \leq (\text{no. of non-eq. pairs}) * \sup_{x,y} \{E[p_{xy}(r)]\}.$$

4. The augmentation function $f(\cdot)$ is a concave function which increases an entry in the matrix from the initial (tabula rasa) value p_1 to a value according as the strength of belief of equivalence of the corresponding pair of words increases. The r-th iterate of this function which enters the estimate of the rate of convergence behaves asymptotically like the function $1-ab^r$, as will be shown below; such a choice is natural for the function which is to indicate increasing strength of belief in the word class equivalence of words. Let $f^{r*}(x)$ denote the r-th iterate of the function $f(x)$ which maps the interval $[0,1]$ into itself; moreover, assume that $f(x)$ is continuous, that $f'(x)$ exists in $(0,1]$, $f(x)>x$, and $f(1)=1$. Then by the successive application of the mean value theorem,

$$
\begin{aligned}
f^{r+1*}(x) = f[f^{r*}(x)] &= f[f^{r*}(x)] - f(1) + f(1) \\
&= 1 - \{f(1) - f[f^{r*}(x)]\} \\
&= 1 - k_r[1 - f^{r*}(x)] \\
&= 1 - k_r\{1-[1-k_{r-1}(1-f^{r-1*}(x))]\} \\
&= 1 - k_r k_{r-1}[1-f^{r-1*}(x)] \\
&\ \ \vdots \\
&= 1 - (k_r k_{r-1} \ldots\ldots k_1)[1-f(x)] \\
&= 1 - (\prod_{i=o}^{r} k_i)(1-x),
\end{aligned}
$$

where the k_i are constants (values of the derivative of $f(\cdot)$ at the appropriate intermediate value).

Now $p_1 > 0$, so $x_1 = f(p_1) = 1-f'(\xi_o)(1-p_1)$, where $\xi_o \in (p_1,1)$.
Denote $f'(\xi_r)$ by k_r, p_1 by x_o, and $f'(1)$ by κ where we assume $0 < \kappa < 1$.

Then $k_o = \dfrac{1-x_1}{1-x_o} < 1$.

In the same way,

$$x_2 = f(x_1) = 1-k_1(1-x_1)$$
$$\vdots$$
$$x_{r+1} = f(x_r) = 1-k_r(1-x_r)$$

where $k_r = \dfrac{1-x_{r+1}}{1-x_r} < 1$.

Now let $K = \prod_{i=o}^{r} k_i$

$$k_i = k_i+\kappa-\kappa = (k_i-\kappa)+ \;= \kappa[1 + \frac{k_i-\kappa}{\kappa}].$$

Denote $\prod_{i=o}^{r} (1 + \frac{k_i-\kappa}{\kappa})$ by $\eta(r)$. Then $K = \kappa^{r+1}\eta(r)$ and

$$\frac{1-f^{r+1*}(x_o)}{\kappa^{r+1}} = (1-x_o)\eta(r).$$

Assume that f' satisfies $|f'(x)-f'(y)| < \lambda|x-y|$ for some constant λ; then $|k_r-f'(1)| = |k_r-\kappa| < |x_r-1|$, for all r; hence the series $\Sigma k_i-\kappa$ is dominated by $\Sigma 1-x_i$ and the latter converges since $k_r < 1$ (i.e., the ratio test). Thus $\eta(r)$ converges to a value which we denote by η.

This proves that

$$f^{r+1*}(p_1) \sim 1 - \eta\kappa^{r+1}.$$

5. Suppose that we center attention on a fixed pair of non-equivalent words x and y. After t sentences in the partitioning procedure have been processed, suppose that k of them produced this pair for comparison: then the x,y entry in the matrix

$$p_{xy}(t) = p(t) = \begin{cases} f^{k*}(p_1) \\ \text{or} \\ 0 \end{cases} ,$$

where $f^{k*}(\cdot)$ denotes the k-th iterate of the augmentation function. This entry is non-zero in the case that k trials took place with x and y believed equivalent; each such trial has probability ε, $\varepsilon = \varepsilon_{xy}$, where epsilon was described in section 2. Hence,

$$E[p(t)\text{: of which k trials involve x,y}] = f^{k*}(p_1)\varepsilon^k,$$

where $0 \leq k \leq t$.

6. Recall that c denotes the probability that words x and y are brought forth for comparison; then for the t trials performed, we have

$$E[p(t)] = (1-c)^t + tc(1-c)^{t-1}\varepsilon f(p_1) + \dots + c^t \varepsilon^t f^{t*}(p_1)$$

$$= \sum_{k=o}^{t} \binom{t}{k} c^k (1-c)^{t-k} \varepsilon^k f^{k*}(p_1)$$

$$\sim \sum_{k=o}^{t} \binom{t}{k} (c\varepsilon)^k (1-c)^{t-k} [1-a\ b^k]$$

$$\sim [(c\varepsilon) + (1-c)]^t - a[(c\varepsilon b) + (1-c)]^t$$

$$\sim [(c\varepsilon) + (1-c)]^t, \quad \text{since } 0 < b < 1;$$

thus,

$$E[p(t)] \sim [1-c(1-\varepsilon)]^t.$$

7. For a given pair of words which is equivalent, after t sentences the x,y entry will be

$$p_{xy}(t) = p(t) = f^{k*}(p_1)$$

for k sentences (out of the t) which involve this pair. Then the mean value of the entry is

$$E[p(t)] = \Sigma_{k=o}^{t} \binom{t}{k} c^k(1-c)^{t-k} f^{k*}(p_1)$$

$$\sim 1 - a[(cb) + (1-c)]^t.$$

8. To summarize, after t sentences have been generated,

the corresponding entry in this "incidence matrix" has mean value for large number of trials t given by

$$E[p_{xy}] \sim \begin{cases} 1 - a[1-c(1-b)]^t & \text{if } x \equiv y \\ [1-c(1-\varepsilon)]^t & \text{otherwise.} \end{cases}$$

9. Rate of Convergence.

A. The syntax-controlled probability sentence generator is a randomly-based mechanism which produces sentences S in L(G)

$$S_1, S_2, \ldots, S_t, \ldots .$$

These events are used to determine word classes. Consider non-equivalent words x and y; the following describes quantities which estimate the rate of convergence to the sought-for partition. We simplify the algorithm as follows: consider the family of events $\{E_a(x,y), a \in V^*\}$, where the $E_a(x,y)$ denote tests of equivalence and each has value 0 or 1. The set V^* can be partitioned into

$$A_{xy} = \{a \in V^* : E_a(x,y) = 1\}$$

and

$$A_{xy}^c = \{a \in V^* : E_a(x,y) = 0\} .$$

Note that A_{xy} includes in its definition string events which do not contain x.

Since $x \neq y$ we are hopeful that an $E_a(x,y)$ might occur with $a \in A_{xy}^c$; but our reality is that it occurs with probability which we denote by π, where π depends on the pair x,y:

$$\mathscr{P}\{E_a(x,y) : a \in A_{xy}\} = \pi$$

so that for the desired events

$$\mathscr{P}\{E_a(x,y) : a \in A_{xy}^c\} = 1-\pi .$$

The events occur (t=1,2,...) and we can tabulate

sentence number	probability of success
1	$1-\pi$
2	$\pi(1-\pi)$
3	$\pi^2(1-\pi)$
$\vdots$	
t	$\pi^{t-1}(1-\pi)$.

The mean sentence number for success is given by

$$1\cdot(1-\pi) + 2\cdot\pi(1-\pi) + \ldots + t\cdot\pi^{t-1}(1-\pi) + \ldots =$$

$$(1-\pi)\,\Sigma_t\, t\pi^{t-1} = (1-\pi)\cdot\frac{1}{(1-\pi)^2} = \frac{1}{1-\pi}.$$

The probability π consists of two parts: the probability that a sentence does not contain the word x and the probability ε_{xy}, where epsilon was described earlier. In the next section we derive formulas for these quantities.

B. To compute π proceed as follows:

define $A_o = \{S \in V^*: S \in L,\ S \text{ does not contain } x\}$

$A_k = \{S \in V^*: \forall S \in L, |S| \geq k \text{ and first occurrence of } x \text{ is at position } k\}$,

where $|S|$ denotes the length of a string S.

Then $L = A_o \cup \bigcup_{k=o}^{\infty} A_k$.

For any string S in A_o, the probability attached to this string is

$$\mathcal{P}\{S: S \in A_o\} = \Sigma_{k=1}^{\infty}\ \mathcal{P}\{S: S \in A_o,\ |S| = k\}.$$

Denote by $P(\zeta) = \{p_{ij}(\zeta)\}$ the probability associated with the continuing rewrite rules $i \to \zeta j$, and by $r(\zeta) = \{r_i(\zeta)\}$ the probability associated with the terminating rules $i \to \zeta$.

Let $d = [1,0,0,\ldots,0]$,

$$\tilde{P} = \left[\sum_{\substack{\zeta\neq x\\ \zeta\in V_T}} p_{ij}(\zeta)\right] \text{ and } \tilde{r} = \left[\sum_{\substack{\zeta\neq x\\ \zeta\in V_T}} r_i(\zeta)\right].$$

Then $$\mathscr{P}\{S:S\in A_o\} = d^T(\tilde{r}+\tilde{P}\tilde{r}+\tilde{P}^2\tilde{r}+\ldots)$$

$$= d^T(I-\tilde{P})^{-1}\tilde{r}$$

as developed in Grenander [1967].

To estimate epsilon, introduce a transformation $\mathscr{T}_k : V^* \to V^*$ defined for strings $|S| \geq k$ as the string $\mathscr{T}_k(S)$ formed by replacement of the k^{th} word in S by the word y.

Define $\quad B_k = \{S:\mathscr{T}_k S \in L, S \in A_k\},\ k \geq 1$

and $\quad B_o = A_o$.

Then these sets can be identified as precisely those which contain strings which cannot separate words x and y. The computation of the probability attached to a string in B_k, which we write as

$$\mathscr{P}\{S:S\in B_k\} = \mathscr{P}\{g(S) = g(\mathscr{T}_k S),\ S \in A_k\}$$

is carried out with the introduction of the following:

let

$$\tau_i(x) = \begin{cases} 1 & r_i(x) > 0 \\ 0 & \text{else} \end{cases} \quad \text{and} \quad t_{ij}(x) = \begin{cases} 1 & p_{ij}(x) > 0 \\ 0 & \text{else} \end{cases}$$

Then for $\mathscr{P}(B_1)$ =

$$r_1(x)\tau_1(y) + \Sigma_\zeta \Sigma_{i_1,j_1} p_{1i_1}(x) r_{i_1}(\zeta_2) t_{1j_1}(y) \tau_{j_1}(\zeta_2) +$$

$$+ \Sigma_\zeta \Sigma_{\substack{i_1,i_2\\ j_1,j_2}} p_{1i_1}(x) t_{1j_1}(y) p_{i_1 i_2}(\zeta_2) t_{j_1 j_2}(\zeta_2) \cdot r_{i_2}(\zeta_3) \tau_{j_2}(\zeta_3) +$$

$$\vdots$$

$$+ \Sigma_\zeta \Sigma_{\substack{(i)\\ (j)}} p_{1i_1}(x) t_{1j_1}(y) p_{i_1 i_2}(\zeta_2) t_{j_1 j_2}(\zeta_2) \ldots \, p_{i_{L-2} i_{L-1}}(\zeta_{L-1}) t_{j_{L-2} j_{L-1}}(\zeta_{L-1}) \cdot r_{i_{L-1}}(\zeta_L) \cdot \tau_{j_{L-1}}(\zeta_L) + \ldots,$$

where the multi-index $(i) = (i_1, i_2, \ldots, i_{L-1})$.

Introduce the tensors

$$S(x,y) = [p_{ij}(x) t_{ik}(y)]$$

$$c(x,y) = [r_i(x) \tau_i(y)]$$

$$M = \Sigma_\zeta \; P(\zeta) \otimes T(\zeta) = [\Sigma_\zeta \; p_{ik}(\zeta) t_{j\ell}(\zeta)]$$

and $$N = \Sigma_\zeta \; r(\zeta) \otimes \tau(\zeta) = [\Sigma_\zeta \; r_i(\zeta) \tau_j(\zeta)],$$

where $\otimes$ indicates the Kronecker product.

Then

$$\mathscr{P}\{S : S \in B_1\} = d^T \cdot [c(x,y) + S(x,y) \Sigma_{L=2}^{\infty} M^{L-2} N]$$

In like manner,

$$\mathscr{P}\{S:S\in B_2\} = d^T[\tilde{P}c(x,y)+\tilde{P}S(x,y)\cdot \Sigma_{L=3}^{\infty} M^{L-3}N]$$

$$\vdots$$

$$\mathscr{P}\{S:S\in B_k\} = d^T[\tilde{P}^{k-1}c(x,y)+\tilde{P}^{k-1}S(x,y)\ \Sigma_{L=k+1}^{\infty} M^{L-(k+1)}N]$$

Since $\Sigma_{L=k+1}^{\infty} M^{L-(k+1)}N = (I-M)^{-1}N$ for $k = 1,2,\ldots$

we get

$$\mathscr{P}\{S\in B_1 \text{ or } S\in B_2 \text{ or } \ldots \text{ or } S\in B_k \text{ or } \ldots\} =$$

$$= d^T\ \Sigma_{k=o}^{\infty}\ \tilde{P}^k[c(x,y)+S(x,y)(I-M)^{-1}N]$$

Using $\Sigma_{k=o}^{\infty}\ \tilde{P}^k = (I-\tilde{P})^{-1}$, we obtain

$$\varepsilon_{xy} = d^T(I-\tilde{P})^{-1}[c(x,y)+S(x,y)(I-M)^{-1}N]/(1-\mathscr{P}(A_o)).$$

The terms and factors can be interpreted as follows:

the i-th of the vector $d^T(I-\tilde{P})^{-1}$ is the probability of getting to state i without writing the word x;

the array $S(x,y)$ selects the appropriate interactions, if any, for a transition from state i to k writing x with probability $p_{ik}(x)$ and from state i to ℓ writing y;

the k,ℓ element of the array $(I-M)^{-1}N$, which is independent of x and y, corresponds to the probability that a path be taken which exits state k and leads to the final state writing the same substring as the one written by some path from state ℓ to the final state.

The factor $(I-M)^{-1}N$, denoted by Q, can be written as

$$Q = MQ+N.$$

By dint of its interpretation as a probability, the diagonal entries

$$q_{ii} = 1;$$

thus Q can be computed from the tensors M and N by iteration. This is important because Q is used to compute the delay in state discovery. Note also that since M is formed from the matrices $P(\zeta)$ and $T(\zeta)$, it is possible to carry out the iterations required without the explicit formation of the tensor M; the non-zero entries of this array can be formed as needed for computation.

From the preceding equations, we get

$$\pi = \Sigma_{k=o}^{\infty} \mathscr{P}(B_k) = d^T(I-\tilde{P})^{-1}[\tilde{r}+c(x,y)+S(x,y)Q].$$

C.1. The calculation of π is carried out for a grammar which is based on the fragment grammar introduced earlier. This new grammar, with fewer states and fewer words, isolates the problems attendent to word class discovery and further reduces the burden of numerical calculations for examples. In addition, this new grammar will help to fix ideas in the chapter.

2. The following "toy grammar" was extracted form the fragment grammar. The 10 word classes (which includes the class DOT containing the singleton word ".") partition the 23 word dictionary. There are 11 state variables: the final state denoted by F and 10 states which correspond to the variables of the grammar governed by 39 rewrite rules which are the expansions of the phrase structure formulas:

$$S \rightarrow NP+AUX+VP1$$
$$NP \rightarrow DET+NA \mid DET+NH$$
$$VP_1 \rightarrow (NOT) + VP+BY+NP1$$
$$NP1 \rightarrow DET+NA \cup NH$$
$$S \rightarrow S+CONJ+S$$

Table 5.1

The rewrite rules are shown below with the probabilities:

1 →	DET	2	1
2 →	NA	3	.5
2 →	NH	4	.5
3 →	AUX	5	1
4 →	AUX	5	.5
4 →	VT	8	.5
5 →	NOT	6	.5
5 →	VP	7	.5
6 →	VP	7	1
7 →	BY	8	1
8 →	DET	9	1
9 →	NH∪NA	10	1
10 →	CONJ	1	.2
10 →	DOT	F	.8

Table 5.2

These expand into the 39 rules as:

Rewrite Rules		Cumulative transition probabilities (for each variable rewritten)
1→11	2	0.3
1→12	2	0.8
1→13	2	1
2→14	3	0.15
2→15	3	0.25
2→16	3	0.4
2→17	3	0.5
2→18	4	0.625
2→19	4	0.75
2→20	4	0.875
2→21	4	1
3→22	5	0.5
3→23	5	1
4→22	5	0.25
4→23	5	0.5
4→27	8	0.75
4→28	8	1
5→32	6	0.5
5→24	7	0.66667
5→25	7	0.83333
5→26	7	1
6→24	7	0.33333
6→25	7	0.66667
6→26	7	1
7→29	8	1
8→11	9	0.3
8→12	9	0.8
8→13	9	1
9→18	10	0.125
9→19	10	0.25
9→20	10	0.375
9→21	10	0.5
9→14	10	0.65
9→15	10	0.75
9→16	10	0.9
9→17	10	1
10→33	F	0.8
10→30	1	0.96
10→31	1	1

Table 5.3

The words in the dictionary are represented above by the integers 11,12,...,33. They are shown below with the "probability within the word class" also shown.

A	11	0.3
THE	12	0.5
SOME	13	0.2
CAT	14	0.3
DOG	15	0.2
KITTEN	16	0.3
PUPPY	17	0.2
BOY	18	0.25
GIRL	19	0.25
MAN	20	0.25
WOMAN	21	0.25
IS	22	0.5
WAS	23	0.5
HELPED	24	0.33333
HURT	25	0.33333
SEEN	26	0.33333
DISLIKES	27	0.5
LIKES	28	0.5
BY	29	1
AND	30	0.8
WHILE	31	0.2
NOT	32	1
.	33	1

Table 5.4

The graph of the corresponding finite state automaton is:

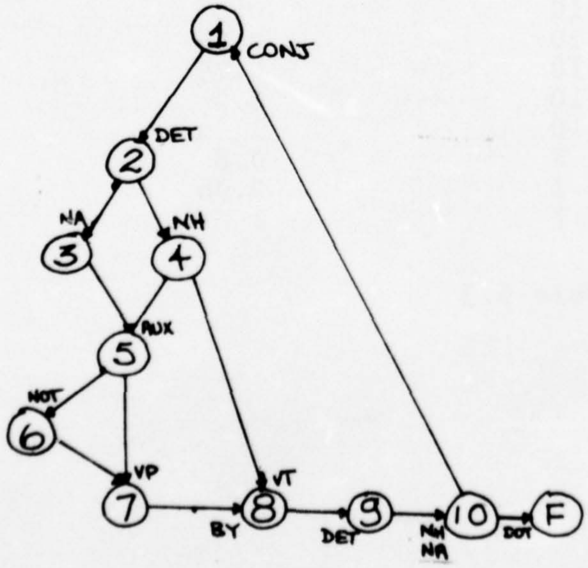


Figure 5.1

3. The matrices $P(\zeta) = \{p_{ij}(\zeta)\}$ consist of 2300 entries, many of which are zero. For any word ζ , $P(\zeta)\otimes T(\zeta)$ is extremely sparse so that M is sparse. The iterative form for the calculation of Q is:

$$Q^{(\rho+1)} = MQ^{(\rho)}+N, \quad \rho=1,2,\ldots$$
$$Q^{(0)} = I$$

where I denotes the 10x10 identity matrix. This simultaneous linear system of equations in the 90 unknowns for q_{ij}, $i\neq j$ can be solved directly, for the example in this section. We observe that the iterations converge because the eigenvalues of M have moduli less than one.

The only non-zero entries in M are listed below:

Indices i	j	k	ℓ	$m_{ijk\ell}$
1	2	1	2	1
1	2	8	9	1
2	3	2	3	1/2
2	4	2	4	1/2
2	3	9	10	1/2
2	4	9	10	1/2
3	5	3	5	1
3	5	4	5	1
4	5	3	5	1/2
4	5	4	5	1/2
4	8	4	8	1/2
5	6	5	6	1/2
5	7	5	7	1/2
5	7	6	7	1/2
6	7	5	7	1
6	7	6	7	1
7	8	7	8	1
8	9	1	2	1
8	9	8	9	1
9	10	2	3	1/2
9	10	2	4	1/2
9	10	9	10	1
10	1	10	1	0.2

Table 5.5

The only non-zero entry in N is

$$N = \Sigma_\zeta r_i(\zeta)\tau_j(\zeta) = r_{10}(33)\tau_{10}(33) = 0.8.$$

After simple arithmetic manipulation, the non-zero, off-diagonal entries of Q are

$$q_{34} = 1$$

$$q_{43} = 1/2$$

$$q_{56} = 1/2$$

$$q_{65} = 1.$$

4. In the following sample computations for specified words x and y

let $P_1 = \mathscr{P}\{g(S) = g(\mathscr{T}_k(S)),\ S \in A_k,\ k \geq 1\}$.

Suppose x = {a}; if y = {a}, then P_1 = 56.5%.

$\mathscr{P}\{S \in A_k,\ k \geq 1\} = 1-\mathscr{P}(A_o) = 1-0.435 = 56.5\%$.

Hence we compute π and find

$\mathscr{P}$\{S:S does not separate x and y\} = 1.

If y = {cat}, then P_1=0 and

$\mathscr{P}$\{S:S does not separate x and y\} = 0.

Suppose x = {cat}; then $\mathscr{P}(A_o)$ is calculated as

$\mathscr{P}$\{S $\in$L:S does not contain {cat}\} = 67.6%.

If y = {cat}, then P_1 = 32.4% and

$\mathscr{P}$\{S:S does not separate x and y\} = 1.

If y = {boy}, then P_1 = 32.4% and

$\mathscr{P}$\{S:S does not separate x and y\} = 1.

Also, $\mathscr{P}\{\mathscr{T}_k(S) \in L:\ S \in A_k,\ k \geq 1\}$ = 32.4% and hence for these x,y
$\varepsilon = .324/(1-.676) = 1$.

Suppose x = {boy};

$\mathscr{P}$\{S $\in$L:S does not contain {boy}\} = 72.3%

If y = {cat}, then P_1 = 20.3% and

$\mathscr{P}$\{S:S does not separate x and y\} = 92.6 %.

That is, for the toy grammar the mean sentence number of success is given by 1/(1-92.3%) = 13. To relate this to the experiment

performed, note that the average sentence length is 9.844 and "boy" is the prototype of a class with seven other words. Then

$$\mathscr{P}\{\{\text{cat}\} \wedge \{\text{boy}\} \wedge S \text{ separates}\} = \mathscr{P}\{S\}\mathscr{P}\{\text{boy}|S\}\cdot\mathscr{P}\{\text{cat}|\text{boy} \wedge S\}$$

A crude estimate for this probability is

$$= (0.077)\left(\frac{1}{9.844}\right)\left(\frac{1}{7}\right)$$
$$= 0.001117432.$$

This small probability indicates that the waiting time for the separation of these two words is long.

Chapter 6

The State Discovery Algorithm

0. In the previous chapter, the word classes were determined by an "interactive" algorithm; that is, an algorithm which depends upon a training sequence establishes the partition of the dictionary of words. The algorithm is based on the principle that an equivalence relation defined on a set induces a partition into equivalence classes. In the case of states, an analogous situation is established by the following observation. The finite state grammar G defined on the word dictionary V_T induces a finite index equivalence relation (EQ) on V_T^* defined as follows: Let u,v in V_T^*; if $g(uz) = g(vz)$ for all $z \in V_T^*$, then uEQv, where $g(\cdot)$ is the grammaticality function introduced earlier. This equivalence relation defines "initial string" equivalence classes which partition V_T^* into the states of the sought for automaton.

1. The synthesis of this completely specified, deterministic finite state automaton is carried out by a process which is modeled after the word class discovery process as closely as is possible. The differences are noted in the next section. REWRITE produces a string in L(G). A transduction of this string replaces each word by its word class prototype and the resultant string of prototypes is processed from left to right. Each string is produced by a sequence of transitions beginning with the distinguished "start" state ($V \leftarrow 0$) to the final state. The <u>tabula rasa</u> hypothesis is that this start state is the *only* state of the automaton. The algorithm decomposes this state into the states of the sought-for automaton. The empty string NULL is the representor of this state; VPROTOS is a list introduced to

keep track of the state representors. The algorithm begins with the start state and its successor word (the first word of the current string); the initial string represented by this pair is classified to a state of the partition under construction. The "target" state is noted and the state equivalence class of this state and its successor word is determined, and so on in this fashion. A graphical overview of the algorithm is given below in flow diagram form:

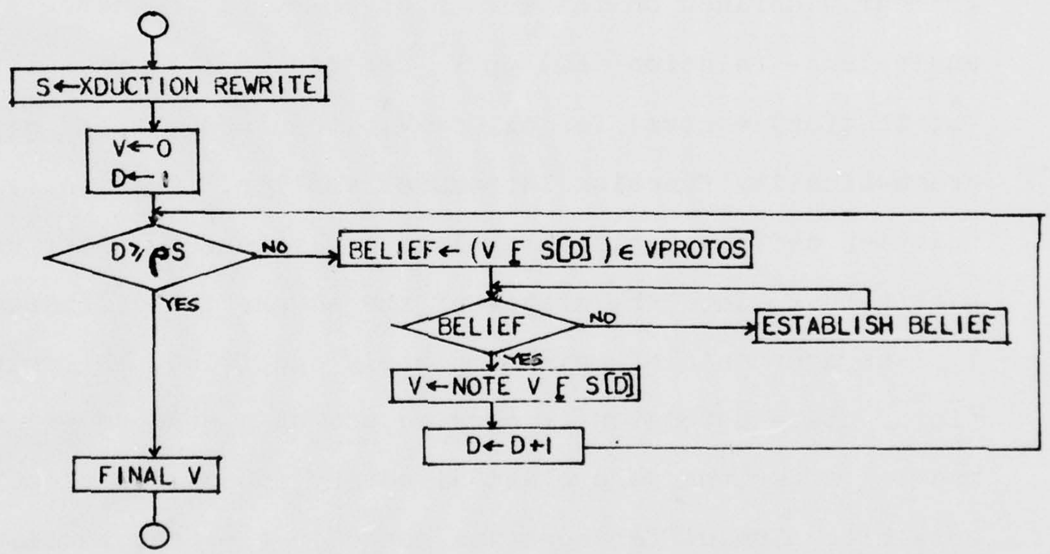


Figure 6.1

2.a. The algorithm is based on the observation that:
If g(uz) ≠ g(vz) for any strings u,v and some z, then u and v are not equivalent. As in the case with word class partitioning, we begin with one equivalence class (V ← 0) and the algorithm will decompose this state into the states of the automaton as required; also, we use the concept of a fixed prototype for each state equivalence class called a <u>representor</u>. The new states are

introduced into the partition as they are "discovered" (i.e., established by the algorithm).

Each string is analyzed from left to right with a depth pointer D which keeps track of the depth of the construction relative to this current string. The string is completely analyzed; that is, the synthesis is carried out for the entire length of each string from the start state to the final state.

b. The representation of a state V followed by a word S[D] is denoted by V $\underline{F}$ S[D]. Moreover, this pair (V,S[D]) is decoded as an integer to radix NC (the number of discovered word classes). As the construction of the automaton is carried out, this list of non-decimal radix integers can be recursively unwound in an obvious way to relate a string and its production sequence relative to the current belief about the partition. That is to say, the structural derivation of a string is developed by the algorithm directly from the string. This scheme eliminates the need for _a priori_ knowledge of all possible combinations of words which might serve as initial strings to which the partitioning is applied; the algorithm builds a list of initial string codes as they occur; it is this list, denoted by VC, to which this abduction algorithm for state discovery applies.

c. At some stage in the algorithm (i.e., the t-th sentence, depth D and having just established that the initial string S[ιD-1] belongs to state V), suppose that the initial string S[ιD], which we denote by V $\underline{F}$ S[D], has occurred for the first time. Then this pair cannot yet be the representor of any state;

i.e., it cannot be among the elements of the list of state representors (VPROTOS). Therefore, we ESTABLISH BELIEF about this new initial string as follows: either V $\underline{F}$ S[D] is to be classified to an existing state of the partition, or, if it is not equivalent to any of these states, it becomes the representor of a newly CREATEd state.

d. ESTABLISHing BELIEF about an initial string. Suppose that a new initial string being classified is V $\underline{F}$ S[D]. Then to ESTABLISH BELIEF about this initial string we successively replace it by the representors of the existing states beginning with the representor of the start state, which is NULL, and test the resultant string (e.g. NULL, D↓S) to determine the relation of this new initial string to the current, established partition. If a state representor is rejected, then the algorithm will either move the new initial string to another state of the partition or CREATE a new state (if all the existing states are rejected). In either of these two cases, the string is moved to a higher index state. Note that it is possible that for some string $z \in V_t^*$, $g(uz) = g(vz)$, but u and v are not equivalent (and this has yet to be discovered). However, once an initial string has been moved out of a state to a higher index state, it is never moved back to a lower index state.

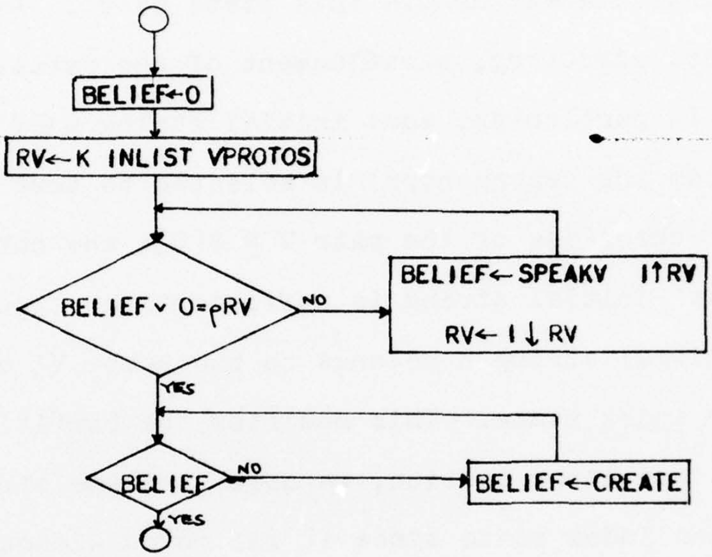
e. To ESTABLISH BELIEF about V $\underline{F}$ S[D] which has already "occurred" at an earlier step (and which is not itself a state representor), replace this initial string by the representor of the state to which it is believed to belong. If this state is rejected, the next, higher index state becomes the candidate for the "target" state to classify this string.

f. If the initial string V $\underline{F}$ S[D] is a state representor, this means that at some earlier stage in the process carried out by this algorithm this pair was established as a new state of the partition into states; denote this state by V'. As in the case of word class discovery, a refinement of the partition might be possible. In particular, some initial string u $\in$ V' (where u is distinct from the representor) is selected to test in the current sentence in the place of the pair V $\underline{F}$ S[D]; the non-grammaticality of the "test" initial string is sufficient to reject the hypothesis that the initial string u belongs to the state V' and u is moved to a higher index state. This modifies the partition. As will be elaborated in a later section, we note that the string u is "tagged" in the higher index state since it was moved without a sentence in which it may have occurred. The tagging will expedite some future selection of this initial string (i.e. when it appears in some future sentence) for classification to a state of the partition.

g. The establishment of states, the classification of initial strings to states, and the code scheme (initial strings as integer codes) determine the partition of V* to within a renumbering of the states of the original automaton. That is, the states and branches are determined. A simple TALLY scheme is used to tabulate the frequency defined probabilities of the transitions between states.

3.a. The flow diagram below depicts the component parts of the process for ESTABLISHing BELIEF. The algorithm is applied to examples in this section to illustrate these steps and to illustrate the algorithm REFINEV which is analogous to the REFINE in word class discovery. In addition, a technique to discover

whether or not loops of a particular kind might occur is introduced; this speeds state discovery at small increase in computation.



ALGORITHM: Establish belief about an initial string
Figure 6.2

b. Consider the grammar over {a,b,c}: 1 → a1, 1 → b2, 2 → b3, 2 → c, 3 → a1, 3 → c, and the corresponding automaton model

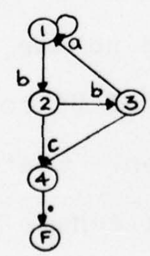


Figure 6.3

where the period, ".", is a symbol added to label the arc to the added final state.

For the purposes of this example, the probabilities are not needed; moreover, the word classes are represented by the letters

themselves. It will be useful to describe the automaton graphically during the various stages of the construction. Initially, it is as shown

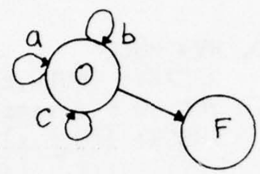


Figure 6.4

The (only) initial state is the start state; the algorithm must decompose or split this state into the sought for states of the automaton. The algorithm will be applied to the training sequence abc, bbc, abbabc.

```
T←1  S←ABC
     V←0
     D←1
          O F A ∈ VPROTOS?
                          NO. RV←NULL
                              SPEAKV: (NULL,BC) ∈ L?
                              YES. O F A TARGETTED TO O
                              CHECK: T(O F A)F A,BC) ∈ L?
                                         I.E., (AABC) ∈ L?
                              YES.
          V←0
          D←2
          O F B ∈ VPROTOS?
                          NO. RV←NULL
                              SPEAKV: (NULL,C) ∈ L?
                                                        NO. CREATE
          V←1
          D←3
          I F C ∈ VPROTOS?
                          NO. RV←NULL B
                              SPEAKV: (NULL) ∈ L?
                                                        NO.
                              SPEAKV:  (B) ∈ L?         NO. CREATE
          V←2
          FINAL

T←2  S←BBC
     V←0
     D←1
          O F B ∈ VPROTOS?
          YES.
          V←1
          D←2
          I F B ∈ VPROTOS?
                          NO. RV←NULL B BC
                              SPEAKV: (NULL,C) ∈ L?
                                                        NO.
                              SPEAKV: (B   ,C) ∈ L?
                              YES.  I F B TARGETTED TO I
                              CHECK: (TI F B)FB,C) ∈ L?
                                     I.E.,(BBB,C) ∈ L?
                                                          NO. REJECT
                              SPEAKV: (BC,C) ∈ L?
                                                        NO. CREATE
```

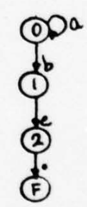


Figure 6.5

```
        V←3
        D←3
        3 F C ∈ VPROTOS?
                    NO. RV←NULL B BC BB
                        SPEAKV: (NULL)∈ L?
                                            NO.
                        SPEAKV: ( B )∈ L?
                                            NO.
                        SPEAKV: (BC )∈ L?
                        YES. 3 F C TARGETTED TO 2
                        CHECK: 3 F C IS ((0 F B)F B)F C
                               2 IS NOT IN THE HISTORY OF 3
                        YES.

        V←2
        FINAL
```

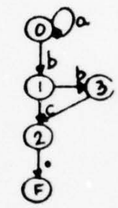


Figure 6.6

```
T←3  S←ABBABC
     V←0
     D←1
        0 F A ∈VPROTOS?
                    NO. RV←NULL B BC BB
                    SPEAKV: (NULL,BBABC)∈ L?
                    YES.  0 F A TARGETTED TO 0
                    CHECK: ((0 F A)F A,BBABC)∈ L?
                           I.E., (AABBABC)∈ L?
                    YES.
        V←0
        D←2
        0 F B∈ VPROTOS?
        YES.
        V←1
        D←3
        1 F B∈ VPROTOS?
        YES.
        V←3
        D←4
        3 F A ∈VPROTOS?
                    NO. RV←NULL B BC BB
                    SPEAKV: (NULL,BC)∈ L?
                    YES. 3 F A TARGETTED TO 0
                    CHECK: 3 F A IS ((0 F B)F B)F A
                           0 IS IN THE HISTORY OF 3
                           ((((((0FB)FB)FA)FB)FB)FA,BC)∈L?
                           I.E., (BBABBABC)∈ L?
                    YES.
```

```
V←0
D←5
0 F B ∈VPROTOS?
YES̄.
V←1
D←6
1 F c∈VPROTOS?
YES̄.
REFINEV:(3 F c)∈L?
        I.E., (BBC)∈L?
YES.
V←2
FINAL
```

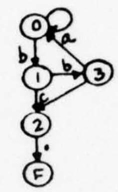


Figure 6.7

c. The function SPEAKV determines whether or not a "trial" target state (a provisional target state) is to be accepted or rejected. That is, suppose that $V_{i_{D-1}}$ $\underline{F}$ S[D] is being classified. Suppose that V'is the state being considered as the target state and that this state has representor V $\underline{F}$ X. Then if the sentence (V $\underline{F}$ X), D↓S is not grammatical, the state V'is rejected as the target state for this initial string. Suppose however that it is grammatical. Then the state $V_{i_{D-1}}$ is unwound to explicitly give the structural derivation of the initial string being classified; this is called the <u>history</u> of this initial string. If the trial target state is among the states in the history of the initial string being classified, then the algorithm can quickly verify that the implied cycle of states is indeed possible. This is done by trying the implied cycle relative to the current initial string and in the current sentence.

If the string associated with the implied cycle cannot be imbedded in the corresponding test string to produce a sentence grammatical in the language L(G), then this is sufficient to reject the trial target state as target to which the initial string is to be classified.

In terms of the worked example above: for the first sentence when 0 $\underline{F}$ a is believed to belong to the trial target state (at D ← 1) by dint of the grammaticality of ((0 $\underline{F}$ a),bc), we note that this implies that - at the very least - ((0 $\underline{F}$ a)$\underline{F}$ a,bc) must also be grammatical. Otherwise, 0 $\underline{F}$ a could not be state equivalent to the state 0. At t ← 2 (D ← 2) 1 $\underline{F}$ b is targeted to state 1. This implies that (1 $\underline{F}$ b)$\underline{F}$ b is a substring which can be imbedded

in the initial string and produce a grammatical sentence. Since it does not, the state 1 is split; that is, despite the grammaticality of (b,c), the state 1 is rejected. In this case, the established states of the current partition were all rejected and so the new state (3) was introduced to the partition.

It is important to note that this technique deals with a special case of the more general problem which confronts the state discovery algorithm and for which the algorithm REFINEV was developed; that is to say, REFINEV is sufficient to carry out the machine synthesis. By checking for these cycles during the classification of an initial string the convergence of the process is improved. In particular, states are discovered earlier; this is important in the early stages of the algorithm because of the frequency tabulations which are later used to estimate the transition probabilities.

d. The second example is based on the output of an inference machine described in Biermann and Feldman [1972]; the grammar is

$1 \to a2$	$2 \to a1$	$3 \to a4$	$4 \to a4$
$1 \to a$	$2 \to b4$	$3 \to a$	$4 \to a$
$1 \to b3$	$2 \to b$	$3 \to b3$	$4 \to b1$

The transition diagram of the corresponding non-deterministic automaton is shown in Figure 6.8.

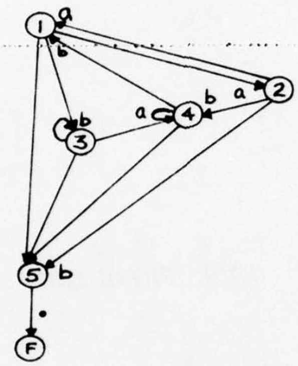


Figure 6.8

The deterministic syntax-controlled transition diagram is shown in Figure 6.9:

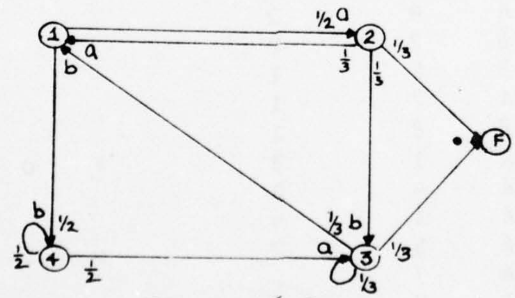


Figure 6.9

the arcs are labeled with the transition probabilities. This is the sought-for automaton.

The notations on the printed output which follows are explained in section 7. For clarity, stages of the automaton under construction are shown in Figure 6.10 (at t=1) and 6.11 (at t=2). The details of the computer implementation follows in section 6. In Figure 6.12, the constructed automaton is shown; the relative frequencies of the internal transitions are also given in parentheses.

1. A B B B B B A A A B B A .

TARGET	S[D]		CODE FREQUENCY
0			
0	A	1	1
0	B	2	1
0	B	2	2
0	B	2	3
0	B	2	4
0	B	2	5
0	A	1	2
0	A	1	3
0	A	1	4
0	B	2	6
0	B	2	7
1	A	1	1

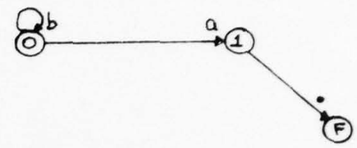


Figure 6.10

2. A A B B B B B A B B A B A A A B A A .

TARGET	S[D]		CODE FREQUENCY
0			
1	A	1	2
0	A	4	1
0	B	2	8
0	B	2	9
0	B	2	10
0	B	2	11
1	B	2	1
0	A	4	2
1	B	2	2
2	B	5	1
2	A	7	1
0	B	8	1
1	A	1	3
0	A	4	3
1	A	1	4
2	B	5	2
2	A	7	2
2	A	7	3

↑	0	2
ω	1	2

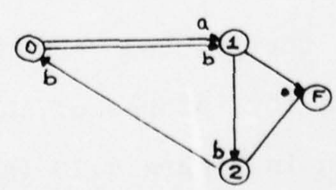


Figure 6.11

3. B A A B A B A A B A A B B B A B A .

TARGET	S[D]		CODE FREQUENCY			
0						
2	B	2	2			
2	A	7	4			
2	A	7	5			
0	B	8	2			
1	A	1	5			
2	B	5	3	ω	6	7
2	A	7	7			
2	A	7	8			
0	B	8	3			
1	A	1	6			
0	A	4	4			
2	B	2	3			
0	B	8	4			
2	B	2	4			
2	A	7	9			
0	B	8	5			
1	A	1	7			

4. A B A .

TARGET	S[D]		CODE FREQUENCY			
0						
1	A	1	8			
2	B	5	4	ω	5	2
2	A	7	10			

5. B B A B A .

TARGET	S[D]		CODE FREQUENCY			
0						
2	B	2	6			
0	B	8	6			
1	A	1	9			
2	B	5	5	ω	7	2
2	A	7	11			

PS 10

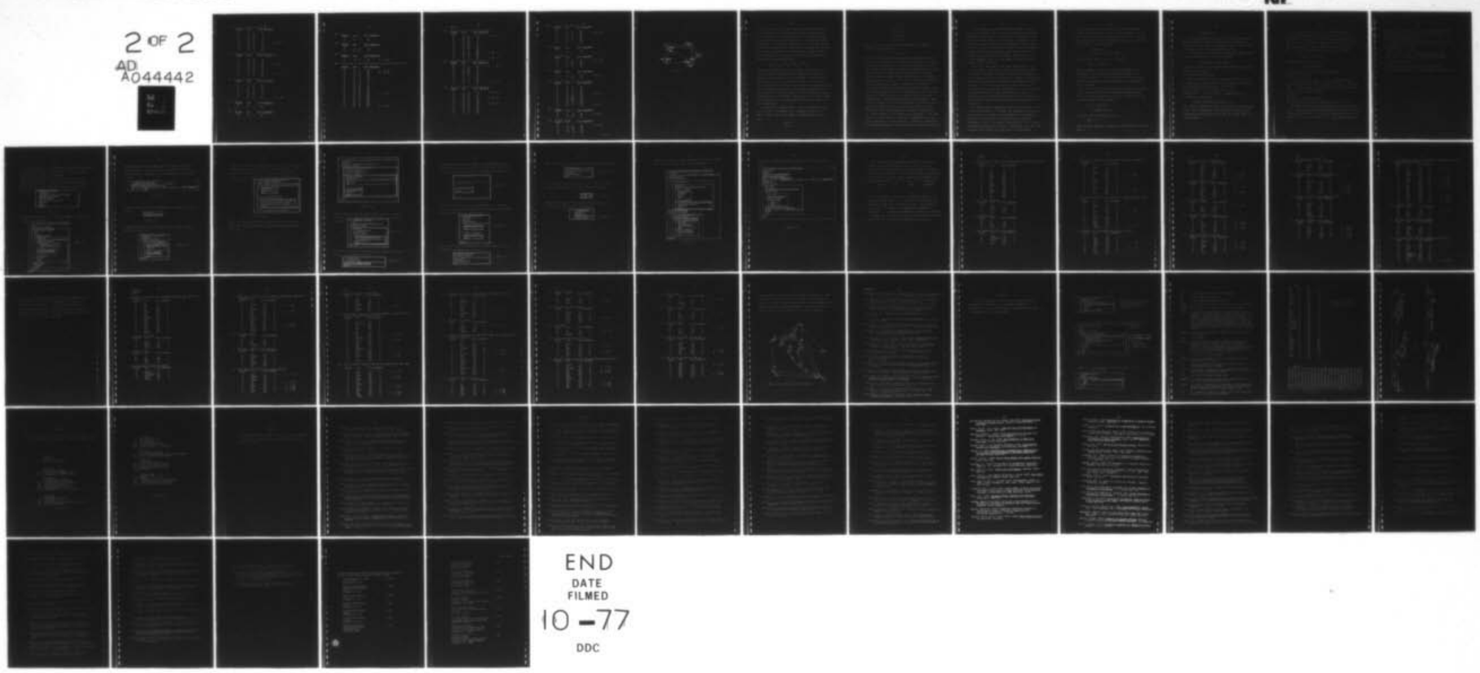
6. B B B A B A B .

TARGET	S[D]		CODE FREQUENCY			
0						
2	B	2	8			
0	B	8	7			
2	B	2	9			
2	A	7	12			
0	B	8	8			
1	A	1	10			
2	B	5	6	+	1	2

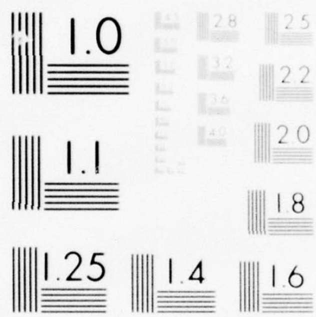
AD-A044 442   BROWN UNIV PROVIDENCE R I DIV OF APPLIED MATHEMATICS   F/G 9/2
ABDUCTION ALGORITHMS FOR GRAMMAR DISCOVERY.(U)
JUN 77 S SHRIER   N00014-77-C-0248

UNCLASSIFIED   NL

2 OF 2
AD
A044442



END
DATE
FILMED
10-77
DDC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

7. *B B B A A B B A A A .*

TARGET	S[D]		CODE FREQUENCY			
0						
3	B	2	2			
0	B	11	1			
3	B	2	3			
0	A	10	1			
1	A	1	11			
2	B	5	7	ω	13	7
0	B	8	9			
1	A	1	12			
0	A	4	5			
1	A	1	13			

8. *B A A B B B B A B A .*

TARGET	S[D]		CODE FREQUENCY			
0						
3	B	2	4			
0	A	10	2			
1	A	1	14			
2	B	5	8	ω	14	7
0	B	8	10			
3	B	2	5			
0	B	11	2			
1	A	1	15			
2	B	5	9	ω	15	7
2	A	7	16			

9. *B B A A B B A A .*

TARGET	S[D]		CODE FREQUENCY			
0						
3	B	2	6			
0	B	11	3			
1	A	1	16			
0	A	4	6			
3	B	2	7			
1	B	11	1			
0	A	4	7			
1	A	1	17	↑	0	11

10. *A .*

TARGET	S[D]		CODE FREQUENCY
0			
1	A	1	18

11. *A .*

TARGET	S[D]		CODE FREQUENCY
0			
1	A	1	19

12. *A* .

TARGET	*S[D]*		*CODE FREQUENCY*
0			
1	*A*	1	20

13. *B A* .

TARGET	*S[D]*		*CODE FREQUENCY*
0			
3	*B*	2	8
1	*A*	10	1

14. *A* .

TARGET	*S[D]*		*CODE FREQUENCY*			
0						
1	*A*	1	21	ω	2	10

15. *A B B A A A A B A A B A B B A A A A A B*
.

TARGET	*S[D]*		*CODE FREQUENCY*			
0						
1	*A*	1	22	↑	0	10
2	*B*	5	10	ω	1	10
0	*B*	8	11			
1	*A*	1	23			
0	*A*	4	8			
1	*A*	1	24			
0	*A*	4	9			
3	*B*	2	9			
2	*A*	10	2			
2	*A*	7	17			
0	*B*	8	12			
1	*A*	1	25			
2	*B*	5	11	↑	0	11
0	*B*	8	13			
1	*A*	1	26			
0	*A*	4	10			
1	*A*	1	27			
0	*A*	4	11			
1	*A*	1	28			
2	*B*	5	12	ω	3	10

PS 10

16. B B B B B B A B A B B A B .

TARGET	S[D]		CODE FREQUENCY			
0						
3	B	2	10	ω	1	11
3	B	11	2			
3	B	11	3			
3	B	11	4			
3	B	11	5			
3	B	11	6			
2	A	10	4			
0	B	8	14			
1	A	1	29			
2	B	5	13	ω	18	7
0	B	8	15			
1	A	1	30			
2	B	5	14	ω	19	7

17. B B B B A A B A B A .

TARGET	S[D]		CODE FREQUENCY			
0						
3	B	2	11	ω	7	11
3	B	11	8			
3	B	11	9			
3	B	11	10			
2	A	10	5			
2	A	7	20			
0	B	8	16			
1	A	1	31			
2	B	5	15	ω	6	10
2	A	7	21			

18. B A B B A B B A A A A A .

TARGET	S[D]		CODE FREQUENCY			
0						
3	B	2	12	ω	11	11
2	A	10	7			
0	B	8	17			
3	B	2	13	ω	12	11
2	A	10	8			
0	B	8	18			
3	B	2	14	ω	13	11
2	A	10	9			
2	A	7	22			
2	A	7	23			
2	A	7	24			
2	A	7	25			

19. *A B A A A A A .*

TARGET	S[D]		CODE FREQUENCY			
0						
1	A	1	32			
2	B	5	16	ω	10	10
2	A	7	26			
2	A	7	27			
2	A	7	28			
2	A	7	29			
2	A	7	30			

20. *A .*

TARGET	S[D]		CODE FREQUENCY
0			
1	A	1	33

21. *B B A A .*

TARGET	S[D]		CODE FREQUENCY			
0						
3	B	2	15	ω	14	11
3	B	11	15			
2	A	10	11			
2	A	7	31			

22. *B A .*

TARGET	S[D]		CODE FREQUENCY			
0						
3	B	2	16	ω	16	11
2	A	10	12			

23. *A A A A B B B A B A B .*

TARGET	S[D]		CODE FREQUENCY			
0						
1	A	1	34			
0	A	4	12			
1	A	1	35			
0	A	4	13			
3	B	2	17	ω	17	11
3	B	11	18			
3	B	11	19			
2	A	10	13			
0	B	8	19			
1	A	1	36			
2	B	5	17	ω	32	7

24. *B A .*

TARGET	S[D]		CODE FREQUENCY			
0						
3	B	2	18	ω	20	11
2	A	10	14			

25. *B B A A A .*

TARGET	S[D]		CODE FREQUENCY			
0						
3	B	2	19	ω	21	11
3	B	11	22			
2	A	10	15			
2	A	7	33			
2	A	7	34			

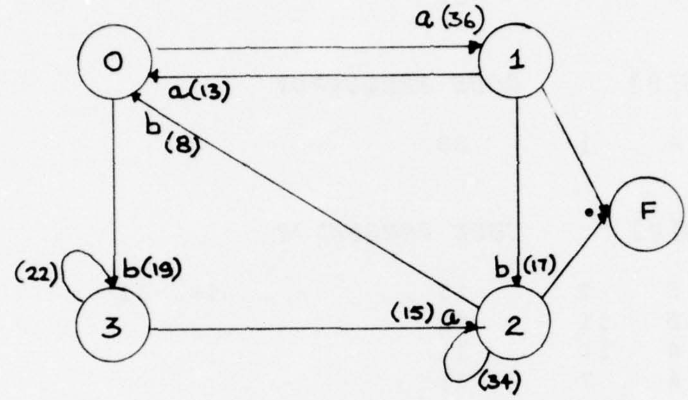


Figure 6.12

4. The constructions carried out above use the representation of the elements being classified (the initial strings) in the form V $\underline{F}$ S[D]; that is, state V which is followed by word S[D]. As noted earlier, the states and words are conveniently denoted by integers. The structural derivation of a string (the sequence of states beginning with the start state and ending with the final state which corresponds to the string) is not normally written with the string. For example, for the string S of length 4, we can show its structural derivation:

$$0=V_0 \;\overset{S[1]}{\quad}\; V_1 \;\overset{S[2]}{\quad}\; V_2 \;\overset{S[3]}{\quad}\; V_3 \;\overset{S[4]}{\quad}\; V_4 \;\overset{\cdot}{\quad}\; F$$

Indeed, the problem developed and solved in this chapter is to find the structural derivations for strings given a sample of strings in the language; the algorithm finds the "intermediate" states $V_1, V_2, \ldots, V_L$ for any string of length L. The explicit form of the next state function can be decoded as a decimal integer which is uniquely associated to the positional numbers V (the state index) and X (the word index) to some radix KEY (an integer greater than one) as "X+V·KEY". The target state to which a code is classified and the code determine the next state function for state V and word X.

In terms of the worked example of section 3b above, let the words a,b, and c correspond to the integers 1,2, and 3; let KEY ← 4 (chosen here as the number of words and "dot" = {·}). Then

$$0 \; \underline{F} \; a \rightarrow 1$$

$$0 \; \underline{F} \; b \rightarrow 2$$

$1\ \underline{F}\ c \rightarrow 7$

$1\ \underline{F}\ b \rightarrow 6$

$3\ \underline{F}\ c \rightarrow 15$

$3\ \underline{F}\ a \rightarrow 13$

That is, the list of codes which correspond to the initial strings of the language is

1 2 6 7 15 13

The codes which correspond to the state prototypes are 0, 2, 6 and 7. For convenience, we included the string NULL (an element of V*) which we decode as 0. In the example above, the sought for automaton was determined to within a renumbering of the states (of the "original" generator/acceptor automaton). The total number of distinct initial string codes which could be formed is sixteen; these are 0, 1, ..., 14, and 15. After the construction has been carried out, a subset of this list of integers is partitioned into the state equivalence classes as follows: state zero contains codes 1 and 13 (for convenience, the code 0 is not included in this list); state one contains code 2; state two contains codes 7 and 15; and state three contains code 6. The remaining "possible" codes (over the alphabet of words a,b, and c and the four states), viz., 3, 5, 9, 10, 11, and 14, are the "discards"; these codes correspond to elements of V* which are not initial strings of any elements of the language L. Note that codes 4, 8, and 12 are not considered as admissible due to the special significance of the symbol ".". Neither the discards nor the inadmissible codes enter the partitioning algorithm explicitly.

It is necessary to select the KEY large enough to ensure that all the possible distinct initial string codes are decoded as distinct integer codes. In the case of a dictionary of NW words, choose KEY as NW. In the case that a language has word classes, a convenient KEY would be the number of these classes (NC) after all the classes have been found. Here and elsewhere, we shall assume that the language has NC classes (where it is possible that NC attains the value NW). Then the number of distinct <u>possible</u> codes (i.e., the number of distinct initial string codes) is (NV)·(NC).

Some of the advantages of this radix representation (over, say, the obvious tabular form) is that (1) Only those decoded values which correspond to initial strings of the language need to get formed; (2) This list structure of integers is partitioned into the state equivalence classes by an algorithm in a manner analogous to the one developed earlier: by decomposition or splitting of classes; (3) The representation of an initial string in "string form" can be easily recovered by the inverse of the decoding function <u>F</u> (described below); and (4) The representation of the initial strings carry their structural derivations within themselves.

The inverse of <u>F</u> is defined as follows: for any code K, and for any integer KEY, K = (V)·KEY+X. The representor of state V, denoted by K', is encoded in the same way: K' = (V')·KEY+X'. Continue in this way until the quotient (in this division algorithm) is zero. The sequence of quotients 0,...,V', V is the structural derivation of the initial string whose code is K. The algorithm which carries out these steps is called ENCODE.

5.a. The syntax-controlled probability sentence generator produces sentences $S_1, S_2, \ldots, S_t, \ldots$ at time intervals $t=1,2,\ldots$ The underlying automaton M has internal states $1,2,\ldots,n_v$, the final state F and a next state function $f: V_N \times V_T \rightarrow V_N$. Denote by $\hat{f}$ the extension of f defined by

$$\hat{f}(V,\mathrm{NULL}) = V$$

$$\hat{f}(1,S) = F \quad \text{for all } S \in L;$$

and for $V \in V_N$, $x \in V_T$, and for all $u \in V_T^*$

$$\hat{f}(V,ux) = \hat{f}(V,u),x).$$

With the addition of the state T which corresponds to the complement of $V_N \cup F$ in V^* the automaton $\hat{M}$ is said to be completely specified. In the following we write M for $\hat{M}$ and f for $\hat{f}$ and we assume that M is deterministic with the minimal number of states.

b. At t=0, there is only one state V (represented by NULL); process each sentence S_t by the algorithm described earlier. That is, for each sentence S(t) at time t of length L(t), and for depth $d=1,2,\ldots|S(t)|$ rewrite

$$\begin{aligned} S(t) &= u_d(t), z_d(t) \\ &= (V\underline{F}X_d(t)), z_d(t). \end{aligned}$$

If for all times t, $z_d(t) \in L$, that is

$$V\underline{F}X_d(t) \equiv 0$$

and convergence obtained. Otherwise, at some finite t, for some depth d

$$0 = V_o\underline{F}X_d(t) \neq 0$$

i.e. (NULL), $z_d(t) \notin L$ and we denote by V_1 the state discovered which is defined by $V_o\underline{F}X_d(t)$ and has as its representor the string denoted $[V_1]$ which is NULL, $X_d(t) = X_d(t)$. This new state corresponds to one of the internal states of M.

c. At any time t suppose m=m(t) states have been discovered and for some depth d=1 or 2 or,..., or $|S(t)|$

$$S(t) = V\underline{F}X, z; \quad \text{then either}$$

(1) $V\underline{F}X$ is classified to an existing state among the discovered states $V_o, V_1, \ldots, V_{m-1}$ or

(2) $V\underline{F}X$ discovers a state V_m.

In order to create this state, we require that $g(u,z) \neq 1$ for all $u \in [V_o, V_1, \ldots, V_{m-1}]$, where $[V_o, V_1, \ldots, V_{m-1}]$ denotes the set of representors of these states. We note that the set $V_o, V_1, \ldots, V_m$ is a subset of $1, 2, \ldots, n_v$. Clearly, $m(t) \leq n_v$ for all times. Otherwise, for some $S = VFX, z \in L$ we would have

$$g(V\underline{F}X, z) \neq g([k], z) \quad \text{for } k=1,2,\ldots,n_v$$

i.e. $V\underline{F}X$ not equivalent to any of the subsets of the partition of V_N. This is impossible. Note that this ensures that the constructed automaton will have at most the minimal number of states required.

d. For any initial string $u = V \underset{\sim}{F} X$ which the algorithm processes by classification to a state, the state index is *non-decreasing*. That is, if for some time t_1, $g(u,z_1) \neq g(\mathrm{NULL},z_1)$, where $S = u,z_1$ then $u \notin \{v:v \equiv \mathrm{NULL}\}$ and it must belong to one of the other subsets. If $g(u,z_1) \neq g([V_1],z_1)$, then u belongs to one of the sets represented by $V_2,\ldots,V_{m(t_1)}$. Suppose

$$g(u,z_1) = g([V_k],z_1).$$

Now suppose that for $t > t_1$, $S = u,z$

$$g(u,z) \neq g([V_k],z);$$

then u belongs to one of the sets represented by

$V_{k+1},V_{k+2},\ldots,V_{m(t)-1}$.

e. The set $\{V_o,V_1,\ldots,V_{m(t)-1}\}$ is a permutation of the indices $\{1,2,\ldots,n_V\}$ for all $t > \tau$, where τ is finite. Otherwise, $m(t) < n_V$ for all time. This can only happen if, for example,

$V \underset{\sim}{F} X,z$ is in L for all strings z

for which

$[V_i],z$ is in L but not conversely.

This results in the incorrect classification of V F X to V_i.

The event $\tilde{S}(t) = [V_i],\tilde{z}$ for which $(V\ F\ X,\tilde{z})$ is not in L must occur. Moreover, the subalgorithm REFINEV requires that selection from the constructed set $\{v:v \equiv [V_i]\}$ be made randomly; hence for finite time t the test event $g(V\ F\ X,\tilde{z})$ occurs with probability one.

This test event will cause V F X to be classified to a higher index class. Hence for some finite time, V F X will be discovered with probability one.

f. The construction produces a deterministic automaton. For otherwise, we would have

$V \underline{F} X \in I = \{v : v \equiv [V_i]\}$ and

$V \underline{F} x \in J = \{v : v \equiv [V_j]\}$ for some V, some X and $V_i \neq V_j$.

Since $V_i \neq V_j$, there exists at least one $z \in V^*$ for which

$$g([I],z) \neq g([J],z).$$

But if $V\ F\ X \in I$, then $g(V\ F\ X,z) = g([I],z)$. Also, if $V\ F\ X \in J$, then $g(V\ F\ X,z) = g([J],z)$. This is a contradiction.

6. The APL description of the algorithm described in the preceding sections is given in this section; it follows a format analogous to the one used in chapter two.

TABULA establishes PROTOS - the word class pointers - to the transduction, initializes VPROTOS, VC, KEY, TALLY and TS, the sentence counter. The random number generator seed is also initialized.

```
∇ TABULA;INDICES
  C←NV+⍳NW
  INDICES←(V1C-0,¯1↓V1C)/⍳⍴V1C
  PROTOS←C[INDICES]
  VPROTOS←VC←,0
  KEY←(1+100×NC),NC←⌈/V1C
  ⎕RL←7*5
  TS←0
  TALLY←,0
```

Figure 6.13

PS processes MORE sentences. It invokes REWRITE to produce the sentence to be processed.

```
 ∇ PS MORE;V;S;D;BELIEF;T1;K;ACTION
  T1←1
NEWS:→OUT IF MORE<T1
    S←XDUCTION REWRITE
    V←0
    D←1
   L1:→THRU IF D≥⍴S
      ACTION←0
      BELIEF←(K←V E S[D])∊VPROTOS
     L2:→L3 IF BELIEF
        ACTION←1
        BELIEF←ESTABLISHV
        →L2
     L3:→L4 IF ACTION
        ACTION←REFINEV
        →L3
     L4:V←NOTE K
      D←D+1
      →L1
   THRU:FINAL V
    T1←T1+1
    →NEWS
OUT:→0
```

Figure 6.14

XDUCTION carries out the transduction: words are replaced by class prototypes. The counter TS is updated and it is printed with the original string and the table headings for output which follows.

```
∇ T←XDUCTION S;NB;CD;B
(B,0ρNB←ρB←CR,(⍕TS←TS+1), '. ' ),PRINT S
T←PROTOS[CD←+⌿(C⍳PROTOS)∘.≤C⍳S]
(6ρ' '), 'TARGET' ,(4ρ' '), 'S[D]' ,(6ρ' '), 'CODE FREQUENCY'
(3ρ' '), 6 0 ⍕0
```

Figure 6.15

F maps the pair V, W into the integer code according as the value of KEY.

```
∇ Z←V F W
Z←KEY⍳V,PROTOS⍳W
```

Figure 6.16

ESTABLISHV carries out the classification of initial strings as described in the text.

```
∇ BELIEF←ESTABLISHV;RV
BELIEF←0
L1:→OUT IF BELIEF
   RV←K INLIST VPROTOS
   L2:→L3 IF BELIEF∨0=ρRV
      BELIEF←SPEAKV 1↑RV
      RV←1↓RV
      →L2
   L3:→L1 IF BELIEF
      BELIEF←CREATE
      →L3
OUT:→0
```

Figure 6.17

For the list of initial string codes which are state representors (VPROTOS) and initial string code K, INLIST returns that sublist of VPROTOS beginning with the state to which K is believed to belong. If K is not yet in the list VC, it is inserted to the initial state (V ← 0).

```
 ∇ Z←K INLIST VPROTOS;IK;Z1;I
L1:→L4 IF(ρVC)≥IK←VCιK
    L2:→L3 IF(1<ρVPROTOS)∨(ρVC)≥IK←VCιK
        VC←VC,K
        TALLY←TALLY,0
        →L2
    L3:→L1 IF(ρVC)≥IK←VCιK
        Z1←¯1+((VCιVPROTOS),1+ρVC)[2]
        VC←VC[ιZ1],K,VC[I←Z1 FρVC]
        TALLY←TALLY[ιZ1],0,TALLY[I]
        →L3
L4:Z←(¯1++/IK≥Z)↓Z←VCιVPROTOS
```

Figure 6.18

SPEAKV: for initial string VC[RJ] substituted in the sentence: reject state if not grammatical, otherwise proceed as described in the text.

```
∇ BELIEF←SPEAKV RJ;RETURN;TARGET;HISTORY;TARGET;U
 BELIEF←0
 RETURN←~ACCEPT(ENCODE VC[RJ]),D↓S
L1:→OUT IF RETURN
   TARGET← 1+(VC⍳VPROTOS)⍳RJ
   HISTORY←¯1↓PATH K
   BELIEF←TARGET∊HISTORY
  L2:→L1 IF RETURN
     L3:→L4 IF~BELIEF
        U←ENCODE K
        RETURN←~ACCEPT U,((¯1+HISTORY⍳TARGET)↓U),D↓S
        BELIEF←0
        →L3

     L4:→L2 IF RETURN
        BELIEF←ADOUBE
        RETURN←1
        →L4

OUT:→0
```

Figure 6.19

ENCODE unwinds an initial string code to the list of string pointers (i.e. to the "subsentence"), relative to the current partition

```
 ∇ Z←ENCODE V;W;FLAG
 Z←⍳0
 FLAG←(V=0)∨(ρVC)<VC⍳V←(,V)[1]
L1:→OUT IF FLAG
    W←KEY⊤V
    Z←,PROTOS[W[2]]
    FLAG←W[1]=0
   L2:→L1 IF FLAG
       W←KEY⊤VPROTOS[1+W[1]]
       Z←(PROTOS[W[2]×⍳W[2]≠0]),Z
       FLAG←W[1]=0
       →L2
OUT:→0
```

Figure 6.20

PATH unwinds an initial string code to the sequence of states relative to the current partition.

```
 ∇ Z←PATH V;W
 Z←(KEY⊤W←V)[2]
BACK:→OUT×⍳0=Z[1]+0×ρZ←(W←KEY⊤W)[1],Z
 →BACK,W←VPROTOS[1+W[1]]
OUT:→0
```

Figure 6.21

The subalgorithm ADOUBE reclassifies an initial string code as required and BELIEF is returned true; if the coders classified correctly, then no action is taken and BELIEF is returned true.

```
∇ BELIEF←ADOUBE;RV

 BELIEF←RJ=1↑RV←K INLIST VPROTOS
L1:→OUT IF BELIEF

   BELIEF←MOVEV
   →L1

OUT:→0
```

Figure 6.22

MOVEV: the code K is moved to a higher state or to the last state (end of the list) as required. The frequency counter TALLY which corresponds to K is reset to zero.

```
 ∇ BELIEF←MOVEV;IK
 BELIEF←0
 IK←VC⍳K
 TALLY[IK]←0
L1:→L2 IF BELIEF∨RJ=¯1↑RV

   BELIEF←HIERV IK
   →L1

L2:→OUT IF BELIEF

   BELIEF←LASTV IK
   →L2

OUT:→0
```

Figure 6.23

HIERV: code K at IK is "promoted" to a higher state; i.e., K at IK is moved rightward in the list VC.

```
∇ Z←HIERV IK;I;R1
R1←RV[1+RV⍳RJ]-1
I←(⍳¯1+IK),(IK F R1),IK,R1 F⍴VC
VC←VC[I]
TALLY←TALLY[I]
Z←1
```

Figure 6.24

LASTV is invoked to move code K at IK to the end of the list VC.

```
∇ Z←LASTV IK;I
I←(⍳¯1+IK),(IK F⍴VC),IK
VC←VC[I]
TALLY←TALLY[I]
Z←1
```

Figure 6.25

F: As in word class partitioning, F cuts a list of B elements at A and returns pointers to elements at the right of A.

```
∇ Z←A F B
Z←A+⍳B-A
```

Figure 6.26

CREATE adds the code K to the initial string representors for the newly established state of the partition.

```
∇ Z←CREATE;IK
IK←VC⍳K
TALLY[IK]←0
VPROTOS←VPROTOS,K
Z←LASTV VC⍳K
```

Figure 6.27

REFINEV tests initial strings classified to a state to ensure that the relation under construction be symmetric.

```
 ∇ ACTION←REFINEV;SZ;VIK;RV;RJ;I;IK;FLAG
 IK←VC⍳K
 SZ←⍳¯1+(-I-(1↓I←VC⍳VPROTOS),1+⍴VC)[VPROTOS⍳K]
 ACTION←0=⍴SZ
L1:→L5 IF ACTION
   L2:→L1 IF ACTION
       VIK←IK+SZ
       FLAG←0=⍴I←(0=TALLY[VIK])/VIK
      L3:→L4 IF FLAG
          K←VC[I[?⍴I]]
          ACTION←1
          FLAG←1
          →L3
      L4:→L2 IF ACTION
          I←,1↑(⌈/TALLY[VIK])-TALLY[VIK]
          K←VC[VIK[⌈/?2⍴⍴I]]
          ACTION←1
          →L4
L5:ACTION←K=VC[IK]
L6:→OUT IF ACTION
     ACTION←ACCEPT(ENCODE K),D↓S
   L7:→L6 IF ACTION
       RV←1↓K INLIST VPROTOS
      L8:→L9 IF ACTION∨0=⍴RV
          RJ←1↑RV
          ACTION←MOVEV
          TALLY[VC⍳K]←¯1
          →L8
      L9:→L7 IF ACTION
          ACTION←CREATE
          →L9
OUT:→0
```

Figure 6.28

Note updates TARGET and produces output to report on the actions taken.

```
∇ TARGET←NOTE K1;K;IK;A;IK1;FLAG;AOK
  K←V E S[D]
  IK←VC⍳K
  TARGET←¯1++/IK≥VC⍳VPROTOS
  TALLY[IK]←TALLY[IK]+1
  A←(1+¯1↑⍴WORDS)↑PRINT,S[D]
  ⎕←(3⍴' '),( 6 0 ⍕TARGET),(6⍴' '),A,' ',( 4 0 ⍕K), 8 0 ⍕TALLY[IK]
  FLAG←K=K1
L1:→OUT IF FLAG
    A← ' ω'
    IK1←VC⍳K1
    AOK←TARGET=¯1++/IK1≥VC⍳VPROTOS
   L2:→L4 IF AOK
       A← ' +'
       AOK←TALLY[IK1]=0
      L3:→L2 IF AOK
          A← ' ↑'
          AOK←1
          →L3
   L4:TALLY[IK1]←TALLY[IK1]+1
    ⎕←(8⍴' '),A,( 4 0 ⍕TALLY[IK1],K1)
    FLAG←1
    →L1
OUT:' '
```

Figure 6.29

7. The state discovery algorithm was applied to the fragment grammar and the results appear below. For each sentence processed, the output consists of the chart shown; it includes the target state followed by a word class prototype (transduction) which decodes to the code indicated and has the frequency in the target state as given in the last column. For illustrations;

target	S[D]	CODE	FREQUENCY
U			
V	X	K	ν

is to be interpreted: $K \leftarrow U\ \underline{F}\ X$ is classifed to state V; its frequency in this state is ν . Note that for any sentence, its transduced string can be read down the chart. The additional information to the right in the chart, when it appears, is information produced when REFINEV is invoked; this consists of a symbol (as in word partitioning), frequency and string code. We note that all the states are discovered at sentence 19.

1. THE DOG IS NOT SEEN BY THE KITTEN AND MARY WAS NOT SEEN BY TH
E KITTEN .

TARGET	S[D]	CODE	FREQUENCY
0			
1	A	1	1
2	CAT	30	1
3	IS	55	1
4	NOT	91	1
5	SEEN	102	1
6	BY	136	1
7	A	139	1
8	CAT	168	1
0	AND	197	1
2	MARY	16	1
3	IS	55	2
4	NOT	91	2
5	SEEN	102	2
6	BY	136	2
7	A	139	2
8	CAT	168	2

2. MARY LIKES THE KITTEN .

TARGET	S[D]	CODE	FREQUENCY
0			
9	MARY	16	1
6	LIKES	218	1
7	A	139	3
8	CAT	168	3

3. IT WAS NOT BLUE .

TARGET	S[D]	CODE	FREQUENCY
0			
10	IT	15	1
11	IS	239	1
12	NOT	275	1
8	BLUE	281	1

4. MARY VIOLENTLY DISLIKES THE WOMAN .

TARGET	S[D]	CODE	FREQUENCY
0			
9	MARY	16	2
13	IMMENSELY	225	1
6	LIKES	310	1
7	A	139	4
8	MAN	167	1

PS 3

5. ROVER IS NOT HELPED BY THE BOY WHILE ROVER WAS NOT SEEN BY SO ME WOMAN .

TARGET	S[D]	CODE	FREQUENCY		
0					
2	TOUKA	17	1		
3	IS	55	3		
4	NOT	91	3		
5	SEEN	102	3		
6	BY	136	3	ω	2 310
7	A	139	5		
8	MAN	167	2		
0	AND	197	2		
2	TOUKA	17	2		
3	IS	55	4		
4	NOT	91	4		
5	SEEN	102	4		
6	BY	136	4	ω	2 218
7	A	139	6		
8	MAN	167	3		

6. SHE DISLIKES THE DOG AND A CHAIR WAS NOT BLUE WHILE JOHN SPEA KS .

TARGET	S[D]	CODE	FREQUENCY		
0					
9	HE	14	1		
6	LIKES	218	3		
7	A	139	7		
8	CAT	168	4	ω	4 167
0	AND	197	3		
1	A	1	2		
10	TABLE	31	1		
11	IS	239	2		
12	NOT	275	2		
8	BLUE	281	2		
0	AND	197	4		
9	MARY	16	3	ω	2 14
8	SPEAKS	219	1		

7. HE CLAIMS THAT JOHN CLAIMS THAT IT WAS NOT BLUE .

TARGET	S[D]	CODE	FREQUENCY		
0					
9	HE	14	3		
14	SAYS	226	1		
0	THAT	342	1		
9	MARY	16	4	ω	4 14
14	SAYS	226	2		
0	THAT	342	2		
10	IT	15	2	ω	2 31
11	IS	239	3		
12	NOT	275	3		
8	BLUE	281	3		

PS 5

8. ROVER WAS NOT SEEN BY THE CAT .

TARGET	S[D]	CODE	FREQUENCY			
0						
2	TOUKA	17	3			
3	IS	55	5			
4	NOT	91	5			
5	SEEN	102	5			
6	BY	136	5	ω	3	310
7	A	139	8			
8	CAT	168	5	ω	2	219

9. MARY WAS HELPED BY A BOY .

TARGET	S[D]	CODE	FREQUENCY			
0						
9	MARY	16	5	ω	5	14
3	IS	216	1			
5	SEEN	79	1			
6	BY	136	6	ω	4	310
7	A	139	9			
8	MAN	167	5			

10. IT WAS NOT GREEN .

TARGET	S[D]	CODE	FREQUENCY			
0						
10	IT	15	3	ω	3	31
11	IS	239	4			
12	NOT	275	4			
8	BLUE	281	4			

11. SOME CAT IS SEEN BY A GIRL .

TARGET	S[D]	CODE	FREQUENCY			
0						
1	A	1	3			
2	CAT	30	2	ω	4	17
3	IS	55	6	ω	2	216
5	SEEN	79	2			
6	BY	136	7	ω	5	310
7	A	139	10			
8	MAN	167	6			

12. ROVER IS NOT HURT BY THE BOY .

TARGET	S[D]	CODE	FREQUENCY			
0						
2	TOUKA	17	5			
3	IS	55	7	ω	3	216
4	NOT	91	6			
5	SEEN	102	6	ω	3	79
6	BY	136	8	ω	6	310
7	A	139	11			
8	MAN	167	7			

13. MARY DISLIKES THE WOMAN .

TARGET	S[D]	CODE	FREQUENCY			
0						
9	MARY	16	6	ω	6	14
6	LIKES	218	4			
7	A	139	12			
8	MAN	167	8			

14. THE SPOTTED KITTEN WAS SEEN BY THE DOG .

TARGET	S[D]	CODE	FREQUENCY			
0						
1	A	1	4			
15	SPOTTED	26	1			
2	CAT	352	1			
3	IS	55	8	ω	4	216
5	SEEN	79	4			
6	BY	136	9	ω	5	218
7	A	139	13			
8	CAT	168	6	ω	3	219

15. THE DESK IS BLUE AND THE YOUNG MAN SINGS .

TARGET	S[D]	CODE	FREQUENCY
0			
1	A	1	5
10	TABLE	31	4
11	IS	239	5
8	BLUE	258	1
0	AND	197	5
1	A	1	6
16	TALL	25	1
9	MAN	374	1
8	SPEAKS	219	4

16. THE DOG IS HELPED BY SOME PUPPY .

TARGET	S[D]	CODE	FREQUENCY			
0						
1	A	1	7			
2	CAT	30	3	ω	2	352
3	IS	55	9	ω	5	216
5	SEEN	79	5			
6	BY	136	10	ω	7	310
7	A	139	14			
8	CAT	168	7	ω	5	219

17. IT WAS GREEN AND ROVER IS NOT HELPED BY SOME PUPPY AND HE WAS NOT SEEN BY THE DOG .

TARGET	S[D]	CODE	FREQUENCY			
0						
10	IT	15	4	ω	5	31
11	IS	239	6			
8	BLUE	258	2			
0	AND	197	6			
2	TOUKA	17	6			
3	IS	55	10	ω	6	216
4	NOT	91	7			
5	SEEN	102	7	ω	6	79
6	BY	136	11	ω	8	310
7	A	139	15			
8	CAT	168	8	ω	6	219
0	AND	197	7			
9	HE	14	7			
3	IS	216	7			
4	NOT	91	8			
5	SEEN	102	8	ω	7	79
6	BY	136	12	ω	9	310
7	A	139	16			
8	CAT	168	9	ω	7	219

18. IT WAS NOT ORANGE .

TARGET	S[D]	CODE	FREQUENCY			
0						
10	IT	15	5	ω	6	31
11	IS	239	7			
12	NOT	275	5			
8	BLUE	281	5			

19. SOME VALUABLE TABLE WAS NOT ORANGE .

TARGET	S[D]	CODE	FREQUENCY
0			
1	A	1	8
17	FINE	27	1
10	TABLE	399	1
11	IS	239	8
12	NOT	275	6
8	BLUE	281	6

20. IT IS ORANGE WHILE THE GIRL IMMENSELY LIKES A DOG .

TARGET	S[D]	CODE	FREQUENCY			
0						
10	IT	15	6	ω	7	31
11	IS	239	9			
8	BLUE	258	3			
0	AND	197	8			
1	A	1	9			
9	MAN	29	1			
13	IMMENSELY	225	2			
6	LIKES	310	10			
7	A	139	17			
8	CAT	168	10	ω	8	219

8. In the following results, we report on an experiment carried out as follows: every word is established as the word class prototype of its own class. For the grammars considered, this establishes NW classes. The state discovery is carried out successfully as shown below. The space complexity of the initial string code list is of course increased.

V1C←152

TABULA
PS 4

1. THE DOG IS NOT SEEN BY THE KITTEN AND MARY WAS NOT SEEN BY TH
E KITTEN .

TARGET	S[D]	CODE	FREQUENCY
0			
1	THE	2	1
2	DOG	74	1
3	IS	131	1
4	NOT	207	1
5	SEEN	237	1
6	BY	310	1
7	THE	314	1
8	KITTEN	385	1
0	AND	452	1
2	MARY	41	1
3	WAS	132	1
4	NOT	207	2
5	SEEN	237	2
6	BY	310	2
7	THE	314	2
8	KITTEN	385	2

2. MARY LIKES THE KITTEN .

TARGET	S[D]	CODE	FREQUENCY
0			
9	MARY	41	1
6	LIKES	500	1
7	THE	314	3
8	KITTEN	385	3

3. IT WAS NOT BLUE .

TARGET	S[D]	CODE	FREQUENCY
0			
10	IT	40	1
11	WAS	548	1
12	NOT	623	1
8	BLUE	637	1

4. MARY VIOLENTLY DISLIKES THE WOMAN .

TARGET	S[D]	CODE	FREQUENCY
0			
9	MARY	41	2
13	VIOLENTLY	514	1
6	DISLIKES	709	1
7	THE	314	4
8	WOMAN	382	1

PS 4

5. ROVER IS NOT HELPED BY THE BOY WHILE ROVER WAS NOT SEEN BY SO
ME WOMAN .

TARGET	S[D]	CODE	FREQUENCY		
0					
2	ROVER	44	1		
3	IS	131	2	ω	2 132
4	NOT	207	3		
5	HELPED	239	1		
6	BY	310	3	ω	2 500
7	THE	314	5		
8	BOY	381	1		
0	WHILE	453	1		
2	ROVER	44	2		
3	WAS	132	3		
4	NOT	207	4		
5	SEEN	237	3	ω	2 239
6	BY	310	4	ω	2 709
7	SOME	315	1		
8	WOMAN	382	2		

6. IT IS NOT ORANGE .

TARGET	S[D]	CODE	FREQUENCY
0			
10	IT	40	2
11	IS	547	1
12	NOT	623	2
8	ORANGE	638	1

7. SHE CLAIMS THAT HE LIKES SOME MAN .

TARGET	S[D]	CODE	FREQUENCY
0			
9	SHE	39	1
14	CLAIMS	516	1
0	THAT	777	1
9	HE	38	1
6	LIKES	500	3
7	SOME	315	2
8	MAN	380	1

8. MARY SPEAKS AND TOUKA IS NOT SEEN BY THE DOG .

TARGET	S[D]	CODE	FREQUENCY		
0					
9	MARY	41	3	ω	2 38
8	SPEAKS	502	1		
0	AND	452	2		
2	TOUKA	43	1		
3	IS	131	3	ω	4 132
4	NOT	207	5		
5	SEEN	237	4	ω	3 239
6	BY	310	5	ω	3 709
7	THE	314	6	ω	3 315
8	DOG	386	1		

PS 3

9\. SHE WAS NOT HELPED BY SOME WOMAN .

TARGET	S[D]	CODE	FREQUENCY			
0						
9	SHE	39	2			
3	WAS	496	1			
4	NOT	207	6			
5	HELPED	239	4			
6	BY	310	6	ω	4	709
7	SOME	315	4			
8	WOMAN	382	3			

10\. HE DISLIKES A CAT AND A MAN IS NOT HELPED BY SOME CAT AND TH
E SPOTTED DOG IS SEEN BY A GIRL .

TARGET	S[D]	CODE	FREQUENCY			
0						
9	HE	38	3			
6	DISLIKES	501	1			
7	A	313	1			
8	CAT	384	1			
0	AND	452	3			
1	A	1	1			
2	MAN	68	1			
3	IS	131	4	ω	2	496
4	NOT	207	7			
5	HELPED	239	5			
6	BY	310	7	ω	4	500
7	SOME	315	5			
8	CAT	384	2			
0	AND	452	4			
1	THE	2	2	ω	2	1
15	SPOTTED	60	1			
2	DOG	802	1			
3	IS	131	5	ω	3	496
5	SEEN	185	1			
6	BY	310	8	ω	2	501
7	A	313	2			
8	GIRL	383	1			

11\. MARY IS SEEN BY SOME CAT AND TOUKA IS NOT HURT BY THE WOMAN
.

TARGET	S[D]	CODE	FREQUENCY			
0						
9	MARY	41	4	ω	4	38
3	IS	495	1			
5	SEEN	185	2			
6	BY	310	9	ω	5	500
7	SOME	315	6			
8	CAT	384	3			
0	AND	452	5			
2	TOUKA	43	2			
3	IS	131	6	ω	4	496
4	NOT	207	8			
5	HURT	238	1			
6	BY	310	10	ω	5	709
7	THE	314	7	ω	3	313
8	WOMAN	382	4			

PS 4

12. SOME VALUABLE CHAIR IS BLUE AND MARY DISLIKES SOME GIRL .

TARGET	S[D]	CODE	FREQUENCY			
0						
1	SOME	3	1			
16	VALUABLE	64	1			
10	CHAIR	857	1			
11	IS	547	2			
8	BLUE	585	1			
0	AND	452	6			
9	MARY	41	5	ω	5	38
6	DISLIKES	501	3			
7	SOME	315	7			
8	GIRL	383	2			

13. THE SHORT BOY SPEAKS .

TARGET	S[D]	CODE	FREQUENCY			
0						
1	THE	2	3	ω	2	3
17	SHORT	58	1			
9	BOY	901	1			
8	SPEAKS	502	2			

14. SHE IS NOT HELPED BY A KITTEN AND HE SAYS THAT TOUKA WAS NOT HELPED BY SOME CAT .

TARGET	S[D]	CODE	FREQUENCY			
0						
9	SHE	39	3			
3	IS	495	2			
4	NOT	207	9			
5	HELPED	239	6			
6	BY	310	11	ω	4	501
7	A	313	4			
8	KITTEN	385	4	ω	2	585
0	AND	452	7			
9	HE	38	6			
14	SAYS	515	1			
0	THAT	777	2			
2	TOUKA	43	3			
3	WAS	132	5			
4	NOT	207	10			
5	HELPED	239	7			
6	BY	310	12	ω	5	501
7	SOME	315	8			
8	CAT	384	4			

15. HE WAS NOT SEEN BY THE DOG .

TARGET	S[D]	CODE	FREQUENCY			
0						
9	HE	38	7			
3	WAS	496	5			
4	NOT	207	11			
5	SEEN	237	5	ω	2	238
6	BY	310	13	ω	6	501
7	THE	314	8	ω	5	313
8	DOG	386	2			

16. A TABLE WAS GREEN .

TARGET	S[D]	CODE	FREQUENCY		
0					
1	A	1	3		
10	TABLE	76	1		
11	WAS	548	2	ω	3 547
8	GREEN	587	1		

17. ROVER WAS HURT BY SOME DOG .

TARGET	S[D]	CODE	FREQUENCY		
0					
2	ROVER	44	3		
3	WAS	132	6		
5	HURT	186	1		
6	BY	310	14	ω	6 709
7	SOME	315	9		
8	DOG	386	3		

18. IT IS GREEN .

TARGET	S[D]	CODE	FREQUENCY		
0					
10	IT	40	3	ω	2 76
11	IS	547	4		
8	GREEN	587	2		

19. JOHN IS NOT HURT BY SOME BOY .

TARGET	S[D]	CODE	FREQUENCY		
0					
2	JOHN	42	1		
3	IS	131	7	ω	6 496
4	NOT	207	12		
5	HURT	238	3		
6	BY	310	15	ω	7 501
7	SOME	315	10		
8	BOY	381	2		

20. ROVER IS HELPED BY THE BOY AND ROVER WAS SEEN BY THE CAT .

TARGET	S[D]	CODE	FREQUENCY		
0					
2	ROVER	44	4		
3	IS	131	8	ω	7 132
5	HELPED	187	1		
6	BY	310	16	ω	8 501
7	THE	314	9	ω	6 313
8	BOY	381	3		
0	AND	452	8		
2	ROVER	44	5		
3	WAS	132	8		
5	SEEN	185	3		
6	BY	310	17	ω	9 501
7	THE	314	10	ω	7 313
8	CAT	384	5		

PS 5

21. IT IS ORANGE .

TARGET	S[D]	CODE	FREQUENCY		
0					
10	IT	40	4	ω	2 857
11	IS	547	5		
8	ORANGE	586	1		

22. IT IS NOT BLUE .

TARGET	S[D]	CODE	FREQUENCY		
0					
10	IT	40	5	ω	3 76
11	IS	547	6		
12	NOT	623	3		
8	BLUE	637	2		

23. IT IS NOT BLUE .

TARGET	S[D]	CODE	FREQUENCY		
0					
10	IT	40	6	ω	3 857
11	IS	547	7		
12	NOT	623	4		
8	BLUE	637	3		

24. ROVER WAS NOT HELPED BY THE CAT .

TARGET	S[D]	CODE	FREQUENCY		
0					
2	ROVER	44	6		
3	WAS	132	9		
4	NOT	207	13		
5	HELPED	239	8		
6	BY	310	18	ω	7 709
7	THE	314	11	ω	8 313
8	CAT	384	6		

25. THE DOG WAS NOT HURT BY THE BOY .

TARGET	S[D]	CODE	FREQUENCY		
0					
1	THE	2	4	ω	3 3
2	DOG	74	2	ω	2 42
3	WAS	132	10		
4	NOT	207	14		
5	HURT	238	4		
6	BY	310	19	ω	10 501
7	THE	314	12	ω	9 313
8	BOY	381	4		

We note that all the states are discovered at sentence 13. The "earlier" discovery in this case (compared to the results in section 7) is due to the altered sequence of random numbers which drives the sentence generator REWRITE. That is, sentence 6 in this section is not the same as sentence six in section 7.

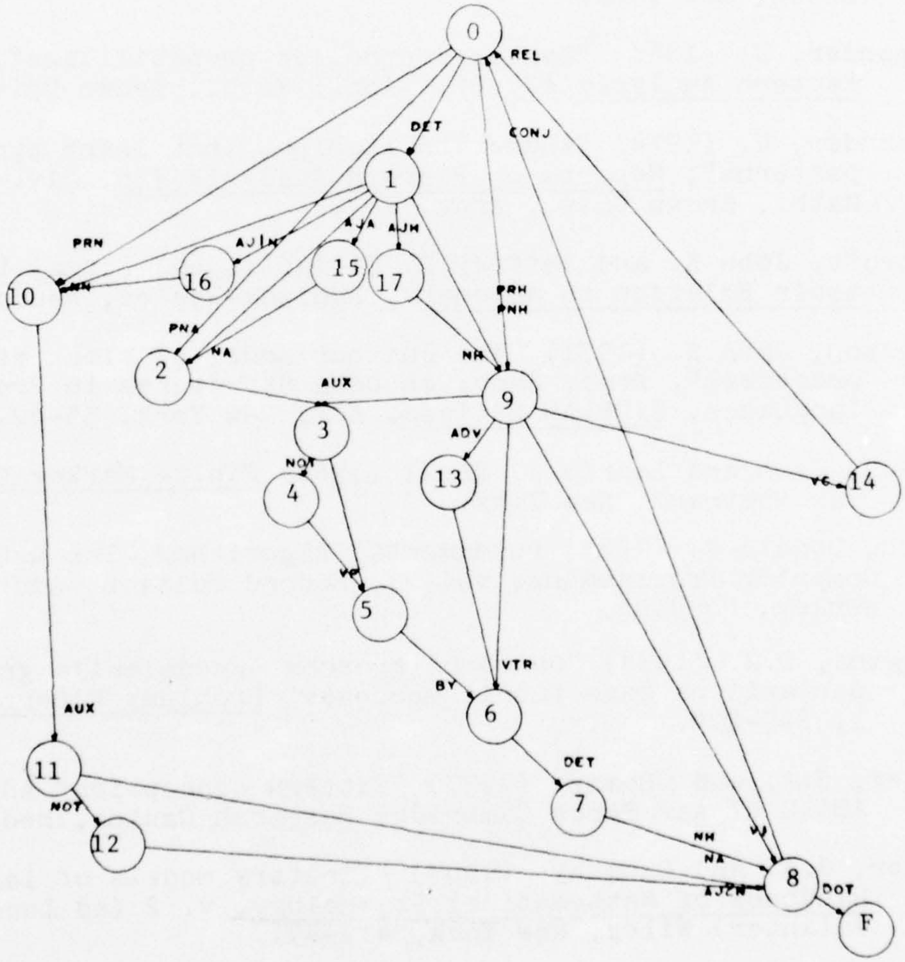


Figure 6.30. The discovered automaton.

Adelman, L. and M. Blum (1975) "Inductive inference and unsolvability", Dept. of E.E., Univ. of Calif., Berkeley (internal document).

Angluin, Dana (1976) "An application of the theory of computational complexity to the study of inductive inference", Memo. ERL-M586, Univ. of Calif., Berkeley, College of Engrg.

Biermann, A.W. and J.A. Feldman (1972) "A survey of results in grammatical inference" in Frontiers of Pattern Recognition, Satosi Watanabe (ed), Academic Press, New York.

Davis, P.J. (1963) Interpolation and Approximation, Blaisdell, Waltham.

Fu, K.S. (1974) Syntactic Methods in Pattern Recognition, Academic Press, New York.

Grenander, U. (1967) "Syntax-controlled probabilities", Reports on Pattern Analysis #2, Div. Appl. Math., Brown Univ., Prov.,R.I.

Grenander, U. (1974) "Abduction machines that learn syntactic patterns", Reports on Pattern Analysis #25, Div. of Appl. Math., Brown Univ., Prov. R.I.

Hopcroft, John E. and Jeffrey D. Ullman (1969) Formal Languages and their Relation to Automata, Addison-Wesley, Reading.

Johnston, John B. (1971) "The contour model of block structured processes", Proc. Symp. on Data Structures in Programming Languages, SIGPLAN Notices, ACM, New York, 55-82.

Kemeny, John and Laurie J. Snell (1960) Finite Markov Chains, van Nostrand, New York.

Knuth, Donald E. (1973) Fundamental Algorithms, The Art of Computer Programming, vol. 1, Second Edition, Addison-Wesley, Reading.

Kulagina, O.S. (1958) "Ob odnom sposobe opredelenija grammaticeskih ponjatii na baze teorii mnozestv" Problemy Kibernetiki, T. 1, 203-214.

Miller, G.A. and Chomsky (1957) "Pattern conception" ASTIA Document AD110 07 Air Force Cambridge Research Center, Bedford.

Miller, G.A. and Chomsky (1963) "Finitary models of language users", Handbook of Mathematical Psychology, v. 2 (ed Luce, Bush, Galanter) Wiley, New York, 419-491.

Peirce, C.S. (1931) The Philosophy of Peirce: Selected Writings, Justus Buchler (ed), Dover, New York, 1955.

Solomonoff, R.J. (1957) "An inductive inference machine", IRE National Convention Record V, part 2, session 22.

Trakhtenbrot, B.A. and Ja. M. Barzdin (1973) Finite Automata: Behavioral Synthesis, American Elsevier.

Appendix A

We collect together in this section descriptions of the subalgorithms referred to in the previous chapters, a descriptive list of the data elements, and hierarchical communication diagrams for the subalgorithms.

```
∇ Z←ACCEPT S;STATE
 STATE←0<ρS←,S
L0:→L1 IF(0=ρS)∨(0=STATE)
    STATE←MATRIX[STATE;S[1]-NV]
    S←1↓S
    →L0
L1:Z←(0=ρS)∧(0=STATE)
```

Figure A.1. For the list of word points S (a sentence), ACCEPT returns the value 1 if S is grammatical, 0 otherwise.

Figure A.2. REWRITE produces a list of word points which corresponds to a sentence in the language.

```
∇ SP←REWRITE;L;I;R;A
 I←1
 SP←,1
L0:→OUT IF I>L←ρSP
  L1:→L2 IF SP[I]∊NV+⍳NW
      A←RULES[CUM[SP[I]]+⍳NRULES[SP[I]];]
      R←(RND≤A[;1])⍳1
      SP←SP[⍳I-1],A[R;2+⍳A[R;2]]
      SP←SP,SP[I-⍳L-I]
      →L1
  L2:I←I+1
   L←ρSP
   →L0
OUT:→0
```

```
∇ R←RND
 R←1E¯6×?1000000
```

Figure A.3. RND returns a random number between 0 and 1.

Figure A.4. PRINT produces the character form of a list of word pointers.

```
∇ S←PRINT SP;I;K
 S←ρK←1
L0:→OUT IF(ρSP)<K
    S←S,' ',WORDS[I;⍳LWORDS[I←SP[K]-NV]]
    K←K+1
    →L0
OUT:→0
```

NW — is the number of words in the vocabulary.

NV — is the number of variables.

CR — is the character "carriage return".

WORDS — is an array of characters of the words in the vocabulary.

INWORDS — is the number of characters in each word in WORDS.

RULES — is an array used by REWRITE to generate sentences in the language. The first column is a list of the cumulative transition probabilities for each variable being rewritten. Across a row in RULES, the rule being applied for the variable is rewritten as the list in column three and four if the entry column two is 2; in the case of a terminating rule column two contains a 1, and only the word in column three is "catenated" to the list of word pointers. In this latter case, column four contains a zero.

NRULES — is a list of the number of rewrite rules available for each variable.

CUM — is 0, NRULES.

MATRIX — is an array which is used by ACCEPT to determine the grammaticality of strings. For each word (first, second,...,last) in the language, the corresponding column in the array MATRIX (i.e., first, second,...,last) is the list of "next states" for that word from the "current state" which is given by the row.

C — is the list of word pointers.

PROTOS — is a list of pointers to the prototypes (as they are discovered) in the list C.

TW — counts the number of sentences processed by PW; it is initialized to zero by BIRTH.

V1C — is the list of indices of each word which corresponds to word classes as determined by PW.

VC — is a list of initial string codes.

VPROTOS — is a list of pointers to the state representors in the list of initial string code list VC.

TALLY — is a list of frequency of occurrence of each string code in the list VC; it tabulates the number of occurrences since classification to a code's current state.

TS — is a sentence counter for number of sentences processed by PS; it is initialized by TABULA.

RULES

0.3	2	11	2
0.8	2	12	2
1	2	13	2
0.15	2	14	3
0.25	2	15	3
0.4	2	16	3
0.5	2	17	3
0.625	2	18	4
0.75	2	19	4
0.875	2	20	4
1	2	21	4
0.5	2	22	5
1	2	23	5
0.25	2	22	5
0.5	2	23	5
0.75	2	27	8
1	2	28	8
0.5	2	32	6
0.66667	2	24	7
0.83333	2	25	7
1	2	26	7
0.33333	2	24	7
0.66667	2	25	7
1	2	26	7
1	2	29	8
0.3	2	11	9
0.8	2	12	9
1	2	13	9
0.125	2	18	10
0.25	2	19	10
0.375	2	20	10
0.5	2	21	10
0.65	2	14	10
0.75	2	15	10
0.9	2	16	10
1	2	17	10
0.8	1	33	0
0.96	2	30	1
1	2	31	1

Figure A.5. RULES is displayed for the toy grammar introduced in Chapter six.

MATRIX

2	2	2	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11
11	11	11	3	3	3	3	4	4	4	4	11	11	11	11	11	11	11	11	11	11	11	11
11	11	11	11	11	11	11	11	11	11	11	5	5	11	11	11	11	11	11	11	11	11	11
11	11	11	11	11	11	11	11	11	11	11	5	5	11	11	11	8	8	11	11	11	11	11
11	11	11	11	11	11	11	11	11	11	11	11	11	7	7	7	11	11	11	11	11	6	11
11	11	11	11	11	11	11	11	11	11	11	11	11	7	7	7	11	11	11	11	11	11	11
11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	8	11	11	11	11
9	9	9	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11
11	11	11	10	10	10	10	10	10	10	10	11	11	11	11	11	11	11	11	11	11	11	11
11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	1	1	11	0
11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11

Figure A.6. MATRIX is displayed for the toy grammar introduced in Chapter six.

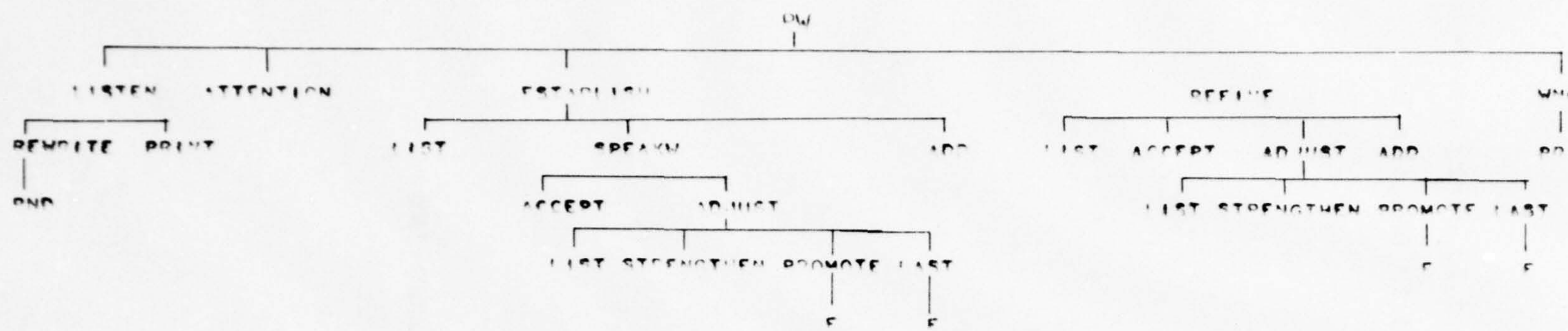


Figure A.6

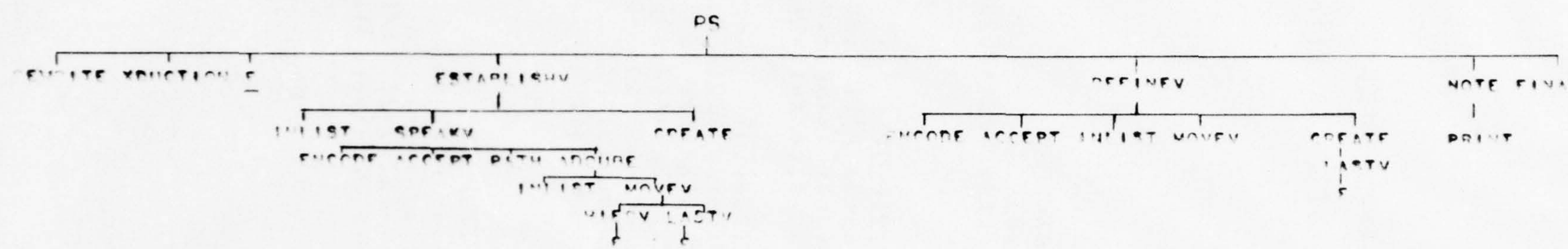


Figure A. 7

Appendix B

The APL programs listed in Figure B.1 were used to carry out the calculations reported and graphically depicted in Chapter four.

```
      ∇ABEC[⎕]∇
    ∇ Z←ABEC X
[1]   Z←P[1]+P[2]×*P[3]×X
    ∇


      ∇FIT1[⎕]∇
    ∇ LINE←ZA FIT1 Z;RL1;M
[1]   M←(RL1←¯1+⍳⍴Z)+.×⍟(ZA-Z)÷ZA-1
[2]   LINE←M,(⍟ZA-1)-M←M÷RL1+.×RL1
    ∇
      ∇FIT1A[⎕]∇
    ∇ NFIT1←ZA FIT1A Z;RES;N
[1]   'X-INTERCEPT IS ';-÷/⌽LINE
[2]   NFIT1←ZA-*((N,1)⍴⍳N←⍴Z)⊥LINE
[3]   RES←(Z-NFIT1)×Z-NFIT1
[4]   'MAXSQRESIDUAL IS ';⌈/RES
[5]   'NORM RESIDUAL IS ';(+/RES)*0.5
    ∇


      ∇FIT2[⎕]∇
    ∇ P←INT FIT2 Z;YL1;R;XI
[1]   R←⍳⍴YL1←Z-1
[2]   P←(-(XI+.×YL1)÷XI+.×XI),0.01 ZERO INT
[3]   P←(1-P[1]×*P[2]),P
    ∇
```

Figure B.1

```
      ∇FIT2A[⎕]∇
    ∇ NFIT2←P FIT2A Z;RES
[1]   NFIT2←ABEC⍳⍴Z
[2]   RES←(Z-NFIT2)×Z-NFIT2
[3]   'MAXSQRESIDUAL IS ';⌈/RES
[4]   'NORM RESIDUAL IS ';(+/RES)*0.5
    ∇
      ∇FN[⎕]∇
    ∇ V←FN C;ERC;ETA;ALPHA
[1]   ALPHA←(ETA←EC-R×ERC)+.×XI←EC-ERC←(EC←*C)*R
[2]   V←YL1+.×ETA-XI×ALPHA÷XI+.×XI
    ∇
      ∇ZERO[⎕]∇
    ∇ Z←TOL ZERO X;G
[1]   →ERR×⍳0<(FN X[1])×FN X[2]
[2]  BACK:→0×⍳TOL≥|G←FN Z←0.5×+/X
[3]   →BACK,X[1+(0≥G×FN X[1])]←Z
[4]  ERR:'ERROR'
    ∇

      ∇LINEFIT[⎕]∇
    ∇ LINE←ZA LINEFIT Z;T;R
[1]   Z←⍟ZA-Z
[2]   LINE←(+/R×Z)-(÷T)×(+/Z)×+/R←⍳T←⍴Z
[3]   LINE←LINE÷(+/R×R)-(÷T)×(+/R)*2
[4]   LINE←LINE,(÷T)×(+/Z)-LINE×+/R
    ∇
```

Figure B.2

Appendix C

The bibliography which follows is not intended to be exhaustive; it is included with the belief that it might prove useful to workers in the area.

Adleman, L. and M. Blum (1975) "Inductive Inference and Unsolvability", Univ. Cal. Berkeley (internal document).

Aiserman, Mark A., et. al. (1971) Logic, Automata & Algorithms, Trans. by Scripta Technica, George M. Kranc (ed.), Academic Press, New York. (Russian Trans. (1963))

Amosov, Nikolai M. (1967) Modeling of Thinking and the Mind, Trans. by Leo Feingold (Trans. Ed., Lawrence J. Fogel), Spartan Books, New York.

Andersen, Henning (1973) "Abductive and Deductive Change", Language, 49(4), Baltimore.

Anderson, John R. (1975) "Computer Simulation of a Language Acquisition System: A First Report", Information Processing and Cognition: The Loyola Symposium, Robert L. Solso Ed., LEA (Wiley), Washington.

Anderson, John R. and Gordon H. Bower (1973) Human Associative Memory, V.H. Winston & Sons (Wiley), Washington, D.C.

Angluin, Dana (1976) "An Application of the Theory of Computational Complexity to the Study of Inductive Inference", Memo. ERL-M586, Univ. of California, College of Engrg., Berkeley.

Arbib, Michael A. (1964) Brains, Machines and Mathematics, McGraw-Hill, New York.

Arbib, Michael A. (1968) Algebraic Theory of Machines, Languages and Semi-Groups, Academic Press, New York.

Arbib, Michael A. (1969) Theories of Abstract Automata, Prentice Hall, Englewood Cliffs.

Arbib, Michael A. (1972) The Metaphorical Brain: An Introduction to Cybernetics as Artificial Intelligence and Brain Theory, Wiley, New York.

Arbib, Michael A. (1974) "Machines in a Category: An Expository Introduction", SIAM Review 16(2), 163-192.

Arbib, Michael A. and Ernest G. Manes (1975) "Adjoint Machines State-behavior Machines and Duality", J. Pure and Applied Algebra 6, 313-344.

Bar-Hillel, Yehoshua (1964) Language and Information: Selected Essays on their Theory and Application, Addison-Wesley, Reading.

Barzdin, J.M. (1970) "On Decoding Automata in the Absence of an Upper Bound on the Number of States", Doklady Akademii Nauki SSSR, t. 190 (5).

Barzdin, J.M. (1972) "On Synthesizing Programs given by examples" Computing Center, Latvian State Univ., Riga, USSR.

Barzdin, J.M. (1972) "Prognostication of Automata and Functions" IFIPS 71, North-Holland, Amsterdam.

Barzdin, J.M. and R.V. Freivald (1972) "On the Prediction of general recursive functions", Soviet Math. Doklady 13(5), 1224-1228, orig: DAN SSSR t. 206(3).

Bellman, Richard and C. P. Smith (1973) Simulation in Human Systems: Decision Making in Psychotherapy, Wiley-Interscience, New York.

Biermann, A.W. (1972) "On the Inference of Turing Machines from Sample Computations", Artificial Intelligence 3(3), 181-198.

Biermann, A.W. and J.A. Feldman (1972) "On the Synthesis of Finite State Machines from Samples of their Behavior", IEEE Trans. Comp. C-21, 592-597.

Biermann, A.W. and J.A. Feldman (1972) "A Survey of Results in Grammatical Inference" in Frontiers of Pattern Recognition Satosi Watanabe, ed., Academic Press, New York, 31-54.

Birkhoff, Garrett and Thomas C. Bartee (1970) Modern Applied Algebra, McGraw-Hill, New York.

Bloom, Lois (1970) Language Development: Form and Function in Emerging Grammars, MIT Press, Cambridge.

Bloom Lois (1973), One Word at a Time: The Use of Single Word Utterances before Syntax, Mouton, The Hague.

Blum, L. and M. Blum (1973) "Inductive Inference: A Recursion Theoretic Approach", IEEE 14th Annual Symposium on Switching and Automata Theory, 200-208.

Blum, L. and M. Blum (1975) "Toward a Mathematical Theory of Inductive Inference", Information and Control 28(2), 125-155.

Booth, Taylor L. and Richard A. Thompson (1973) "Applying Probability Measures to Abstract Languages", IEEE Trans. Comp. C-22(5), 442-450.

Booth, Taylor L. (1969) "Probabilistic Representation of Formal Languages", IEEE Conf. Record 10th Symp. Switching and Automata Theory, 74-81.

Botvinnik, M.M. (1970) Computers, Chess and Long Range Planning, trans. Arthur Brown, Springer-Verlag, New York.

Braffort, P. and D. Hirschberg (1963) eds., Computer Programming and Formal Systems, North-Holland, Amsterdam.

Brayer, John M. and K.S. Fu (1977) "A Note on the k-tail Method of Tree Grammar Inference", IEEE Trans. Systems, Man, and Cybern., SMC-7(4), 293-300.

Brown, Roger (1970) Psycholinguistics: Selected Papers, The Free Press, New York (Collier-Macmillan).

Brown, Roger (1973) A First Language: The Early Stages, Harvard University Press, Cambridge.

Buchler, Justus (ed.)(1955) Philosophical Writings of Peirce, Dover, New York.

Caianiello, E.R. (Ed.)(1966) Automata Theory, Academic Press, New York.

Chandrasekaran, B. (1975) "Artificial Intelligence - The Past Decade", Advances in Computers, v. 13, Academic Press, New York.

Chomsky, Carol (1969) The Acquisition of Syntax in Children from 5 to 10, MIT Press, Cambridge.

Chomsky, N. (1956) "Three Models for the Description of Language", IRE Trans. Information Theory, IT-2, 113-124.

Chomsky, N. (1957) Syntactic Structures, Mouton & Co., The Hague.

Chomsky, N. (1959) "On certain formal properties of grammars", Information and Control, 2(2) 137-167.

Chomsky, N. (1959) "A Note on Phrase Structure Grammars", Inf. & Control, 2(4) 393-395.

Chomsky, N. (1971) "On Interpreting the World", Problems of Knowledge and Freedom, Vintage, New York, 3-51.

Chomsky, Noam (1972) Language and Mind, enlarged edition, Harcourt, Brace Jovanovich, New York.

Chomsky, N. and George A. Miller (1958) "Finite State Languages", Information and Control, 1(2) 91-112.

Chomsky, N. and George A. Miller (1963) "Introduction to the Formal Analysis of Natural Languages" in Handbook of Mathematical Psychology, v. 2, Bush, Galanter, Luce (eds.), Wiley, New York, 269-321.

Cohen, Joel M. (1967) "The Equivalence of Two Concepts of Categorical Grammar", Inf. & Control 10(5) 475-484.

Cohen, Rina and Karl Culik II (1971) "LR-regular Grammars - An Extension of LR(k) Grammars", IEEE Conf. Rec. 1971 - 12th Symp. Switching and Automata Theory.

Cohn, P.M. (1975) "Algebra and Language Theory", Bull. London Math. Soc. 7, 1-29.

Cook, Craig M., Azriel Rosenfeld and Alan R. Aronson (1976) "Grammatical Inference by Hill-Climbing", Information Sciences 10, 59-80.

Crespi-Reghizzi, S. (1971) "An Effective Model for Grammatical Inference", IFIP Congress 1971 Ljubljana, v. 1, North-Holland, Amsterdam, 524-529.

Crespi-Reghizzi, S. (1971) "Reduction of Enumeration in Grammar Acquisition", Proc. Int. Joint Conf. Artificial Intelligence (2nd) London, Xerox University Microfilms, Ann Arbor 1975 CSIBI,

Crespi-Reghizzi, S., M.A. Melkanoff and L. Lichten (1973) "The Use of Grammatical Inference for Designing Programming Languages", CACM 16(2) 83-90.

Csibi, Sandor (1975) Stochastic Processes with Learning Properties, Springer-Verlag, Wien.

Curry, Haskell B. and Robert Feys (1958) Combinatory Logic, North-Holland, Amsterdam.

Dahl, O.-J., E.W. Dijkstra and C.A.R. Hoare (1972) Structured Programming, Academic Press, London.

Davis, Martin (1958) Computability and Unsolvability, McGraw-Hill New York.

Davis, Martin (ed)(1965) The Undecidable, Raven, Hewlett.

Elliot, Lois L. (1975) "Research on Normal Acquisition of Language" in The Nervous System, v. 3 D.B. Tower (Ed.), Raven Press, New York.

Evans, Thomas G. (1971) "Grammatical Inference Techniques in Pattern Analysis", Software Engineering, COINS III, J.T. Tou (ed.), Academic Press, New York.

Even, Shimon (1965) "Comments on the Minimization of Stochastic Machines", IEEE EC-14, 634-637.

Feigenbaum, E.A. (1968) "Artificial Intelligence: Themes in the Second Decade", Proc. IFIPS, Spartan Press, Baltimore.

Feigenbaum, E.A. and J. Feldman (1963) Computers and Thought, McGraw-Hill, New York.

Feldman, J. (1972) "Some Decidability Results on Grammatical Inference and Complexity", Information & Control 20(3) 244-262.

Feldman, J., J. Gips, J.J. Horning and S. Reder (1967) "Grammatical Complexity and Inference", Stanford AI Project, Memo. 55.

Feliciangeli, Horacio and Gabor T. Herman (1973) "Algorithms for Producing Grammars from Sample Derivations", J. Comp. Sys. Sci. 7, 97-118.

Fillmore, Charles (1968) "The case for case", in Universals in Linguistic Theory, E. Bach and R. Harms, (eds), Holt, Rinehart, Winston, New York.

Fischer, Michael J. (1972) "Efficiency of Equivalence Algorithms", Complexity of Computer Computations, R.E. Miller and James W.Thatcher (Eds.), Plenum, New York.

Fodor, J.A. and J.J. Katz (1964)(eds.) The Structure of Language. Readings in the Philosophy of Language. Prentice-Hall, Englewood Cliffs.

Fogel, Laurence J. Alvin J. Owens and Michael J. Walsh (1966), Artificial Intelligence Through Simulated Evolution, Wiley, New York.

Fréchet, Maurice (1952) Recherches Théoriques Modernes sur le calcul des probabilitiés, second libre, deuxièmie édition, Paris, Gauthier-Villars.

Fu, King-Sun (1967) "Stochastic Automata as Models of Learning Systems", Computer and Information Sciences, II Julius T. Tou (ed.) Academic Press, New York, 177-191.

Fu, King-Sun and Taylor L. Booth (1957) "Grammatical Inference: Introduction and Survey", Part 2, IEEE Trans. Systems, Man, and Cybernetics, SMC-5(4), 409-423.

Fu, King-Sun and Taylor L. Booth (1975) "Grammatical Inference: Introduction and Survey", Part I, IEEE Trans. Systems, Man, and Cybernetics, SMC-5(1) 95-111.

Fu, K.S. and T. J. Li (1969) "On Stochastic Automata and Languages", Information Sciences 1, 403-419.

Gardner, Martin (1976) "On the Fabric of Inductive Logic, and Some Probability Paradoxes" in Mathematical Games, Scientific American 234(3) 119-123.

Gerhart, S.L. (1972) "Verification of APL Programs", Ph.D. Thesis, Dept. of Comp. Sci., Carnegie-Mellon, Pittsburgh.

Ginsburg, Seymour (1966) The Mathematical Theory of Context-free Languages, McGraw-Hill, New York.

Ginsburg, S. and Barbara H. Partee (1969) "A Mathematical Model of Transformational Grammar", Inf. & Cont. 15( ) 297-334.

Ginzburg, Abraham (1968) Algebraic Theory of Automata, Academic Press, New York.

Gold, E. Mark (1965) "Limiting Recursion", J. Symbolic Logic, 30(1), 28-48.

Gold, E. Mark (1965) "A Formal Theory of Problem-Solving, in Biophysics and Cybernetic Systems, Proc. 2nd Cybernetic Sciences Symp., M. Maxfield, A. Callahan and L.J. Fogel (eds.) Spartan, Washington, 89-99.

Gold, E. Mark (1967) Language Identification in the Limit, Information and Control 10(5), 447-474.

Grenander, Ulf (1976) Pattern Synthesis: Lectures in Pattern Theory, vol. 1, Springer-Verlag, New York.

Grenander, Ulf (1966) "Can We Look Inside an Unreliable Automaton?", Jerzy Neyman Festschrift, Wiley, New York.

Grenander, Ulf (1967) "Syntax-Controlled Probabilities, Pattern Analysis Report No. 2, D.A.M., Brown University.

Grenander, Ulf (1974) "Abduction Machines that Learn Syntactic Patterns", Pattern Analysis Report No. 25, D.A.M. Brown University.

Gross, Maurice (1972) Mathematical Models in Linguistics, Prentice-Hall, Englewood Cliffs.

Gross, Maurice and A. Lentin (1970) Introduction to Formal Grammars, Springer-Verlag, New York.

Hamburger, H. and K. Wexler (1973) "Identifiability of Transformational Grammars" in Approaches to Natural Language J. Hintikka, J. Moravcik and P. Suppes (eds.), Reidel, Dordrecht, 153.

Hamburger, H. and K. Wexler (1975) "A Mathematical Theory of Learning Transformational Grammar", J. Math. Psych 12, 137-177.

Hammond, Allen A. (1973) "AI: A Fascination with Robots or a Serious Intellectual Endeavor?" SCIENCE, 180, 1352-1353.

Harary, Frank and E. Palmer (1967) "Enumeration of Finite Automata", Information and Control 10(5), 499-508.

Harary, Frank, R.Z. Norman and D. Cartwright (1965), Structural Models, Wiley, New York.

Hart, John F. and Satoru Takasu (Eds.) (1967) Systems and Computer Science, University of Toronto Press, Toronto.

Hartmanis, J. and R.E. Stearns (1966) Algebraic Structure Theory of Sequential Machines, Prentice-Hall, Englewood Cliffs.

Hartshorne, Charles and Paul Weiss (1931-1936) Collected Papers of Charles Sanders Peirce, Harvard University Press, Cambridge.

Hayes, John R. (ed.)(1970), Cognition and the Development of Language, Wiley, New York.

Hedrick, Charles L. (1976) "Learning Production Systems from Examples", Artificial Intelligence, 7, 21-49.

Heerden, Pieter J. van (1968) The Foundation of Empirical Knowledge, Wistik Wassenaar.

Herman, Gabor T. and Grzegorz Rozenberg (1975) Developmental Systems and Languages, North-Holland, New York.

Hermes, H. (1969) Enumerability, Decidability, Computability: An Introduction to the Theory of Recursive Functions, Springer-Verlag, New York.

Hennie, Fred C. (1968) Finite State Models for Logical Machines, Wiley, New York.

Horning, J.J. (1972) "A Procedure for Grammatical Inference", IFIPS 71, C.V. Freiman (ed.), North-Holland, Amsterdam.

Hunt, Earl B. (1975), Artificial Intelligence, Academic Press, New York.

Hunt, Earl B., Janet Marin and Philip J. Stone (1966) Experiments in Induction, Academic Press, New York.

Hunt, Earl B. and C.I. Hovland (1961) "Programming a Model of Human Concept Formation", Proc. West. Joint Computer Conf., 19, 145-155.

Johnston, John B. (1971) "The Contour Model of Block Structured Processes", Proc. Symp. on Data Structures in Programming Languages, SIGPLAN Notices, ACM, New York, 55-82.

Kain, R.Y. (1972) Automata Theory, Machines and Languages, McGraw-Hill, New York.

Kalmár, László (ed.)(1965) Colloquium on the Foundations of Mathematics, Mathematical Machines and their Applications, Akadémiai Kiadó, Budapest.

Kaplan, Ronald M. (1972) "Augmented transition networks as psychological models of sentence comprehension", Artificial Intelligence 3, 77-100.

Kemeny, John G. and J. Laurie Snell (1960) Finite Markov Chains, van Nostrand, Princeton.

Kiefer, Ferenc (1968) Mathematical Linguistics in Eastern Europe, American Elsev., New York.

Kleene, S.C. (1952) Introduction to Metamathematics, van Nostrand, Princeton.

Knobe, Bruce and Katheleen Knobe (1976) "A Method for Inferring Context-free Grammars, Information and Control 31(2) 129-146.

Kobrinski, N.E. and B.A. Trakhtenbrot (1956) Introduction to the Theory of Finite Automata, Trans. by J.C. Sheperdson, North-Holland, Amsterdam.

Kohavi, Zvi (1970) Switching and Automata Theory, McGraw-Hill, New York.

Kohavi, Zvi and Azaria Paz (Eds.)(1971) Theory of Machines and Computations, Academic Press, New York.

Kulagina, O.S. (1958) "A Method of Determining Grammatical Concepts on the Basis of Group Theory" in Bush, Galanter, Luce, Readings. op. cit.

Lambek, Joachim (1958) "The Mathematics of Sentence Structure", Am.Math. Monthly 65, 154-170.

Lee, H.C. and K.S. Fu (1972) "A Syntactic Pattern Recognition System with Learning Capability", Proc. Int. Symp. Comp. and Inform. Sci. (COINS-72).

Lenneberg, Eric H. (1967) Biological Foundations of Language, Wiley, New York.

Lenneberg, Eric H. (1969) "On explaining language" Science, 164(3880), 635-643.

Luce, R.D., R.R. Bush and E. Galanter (eds.)(1963) Handbook of Mathematical Psychology, v. 2, Wiley, New York.

Luce, R.D., R.R. Bush and E. Galanter (eds.)(1965) Readings in Mathematical Psychology, v. 2, Wiley, New York.

Lukasiewicz, Jan (1957) Aristotle's Syllogistic from the Standpoint of Modern Formal Logic, 2nd Edition, Oxford Univ. Press, New York.

Lyons, J. and R.J. Wales (eds.)(1966) Psycholinguistic Papers, Proc. 1966 Edinburgh Conf., Edinburgh Univ. Press, Edinburgh.

Mandelbrot, Benoit (1954) "On Recurrent Noise Limiting Coding", Proc. Symp. on Inf. Networks, Polytech. Press, New York, 205-221.

Marcus, Solomon (1964) Gramatici si Automate Finite, Editura Academiei Republicii Populare Romine, Bucuresti (in Rumanian).

Marcus, Solomon (1967) Alegebraic Linguistics; Analytical Models Academic Press, New York.

Maturana, Humberto (1970) "Biology of Cognition", Biological Computer Laboratory, Elec. Engrg. Dept., Univ. of Illinois, Urbana.

Marynaski, Fred J. (1974) "Inference of Probabilistic Grammars" CS-74-9, Ph.D. Dissertation, Univ. of Connecticut, Storrs.

McNaughton, R. (1961), "The Theory of Automata, A Survey" Advances in Computers, (ed.) Franz L. Alt, Academic Press, New York.

McNeill, David (1970), The Acquisition of Language: The Study of Developmental Psycholinguistics, Harper & Row, New York.

McNeill, David (1975) "The Place of Grammar in a Theory of Performance", Developmental Psycholinguistics and Communication Disorders, D. Aronson and R.W. Rieber (eds.) Annals of the New York Academy of Sciences, New York.

Meyer, A.R. and M.J. Fischer (1971) "Economy of Description by Automata, Grammars, and Formal Systems", IEEE Conf. Record 12th Annual Symp. Switching and Automata Theory.

Miller, George A. (ed)(1973) Communication, Language and Meaning: Psychological Perspectives, Basic Books, New York.

Miller, G.A., E.B. Newman, and E.A. Friedman (1958) Length-Frequency Statistics for Written English, Information and Control 1( ), 370-389.

Millward, R.B. and J.D. Wickens (1975) "Concept Identification Models", Contemporary Developments in Mathematical Psychology: Learning, Memory, and Thinking, vol. 1, D.H. Krantz, et.al. (eds.) W.H. Freeman, San Francisco.

Minsky, Marvin (ed.), (1968) Semantic Information Processing MIT Press, Cambridge.

Minsky, Marvin (1967) Computation: Finite and Infinite Machines, Prentice-Hall, Englewood Cliffs.

Minsky, Marvin (1969) Perceptrons: An Introduction to Computational Geometry, 2nd Printing with corr., MIT Press, Cambridge.

Moore, E.F. (1956) "Artificial Living Plants", Scientific American, 195, October, 118-126.

Moore, E.F. (1959) "Review of Four Papers by Penrose", IRE Trans. Electr. Comput. EC8, 407-408.

Moore, E.F. (ed.)(1964) Sequential Machines: Selected Papers, Addison-Wesley, Reading.

NATO (1971), AGARD: Advisory Group for Aerospace Research and Development, Conf. Proc. 94 on Artificial Intelligence, Rome.

Nelson, R.J. (1968) Introduction to Automata, Wiley, New York.

Nerode, Anil (1958) "Linear Automaton Transformations", Proc. Am. Math. Soc. 9, 541-544.

Nilsson, Nils J. (1965) Learning Machines: Foundations of Trainable Pattern Classification Systems, McGraw-Hill, New York.

Ott, Gene and N. Feinstein (1961) "Design of Sequential Machines from their Regular Expressions", J. Assoc. Comp. Mach. 8(4), 585-600.

Patel, A. R. (1972) "Grammatical Inference for Probabilistic Finite State Languages", Dept. of Elec. Engrg. Univ. of Conn., Storrs.

Pattee, H.H., et.al. (1966) "Natural Automata and Useful Simulations", Proc. Symp. on Fundamental Biological Models, Spartan, Washington.

Paz, Azaria (1971) Introduction to Probabilistic Automata, Academic Press, New York.

Paz, Azaria and Zvi Kohavi (eds.)(1971) Intern. Symposium on the Theory of Machines and Computation, Technion, Haifa.

Peirce, Charles S. (1940) The Philosophy of Peirce: Selected Writings (ed.) Justus Buchler, Harcourt Brace, New York.

Peirce, Charles S. (1966) Selected Writings, P.P. Wiener (ed.) Dover, New York.

Peizer, D.B. and D.L. Olmsted (1969) "Modules of Language Acquisition", Language, 45, 61-96.

Plotkin, G.D. "Inductive Generalization", Machine Intelligence 5, Am. Elsevier, New York.

Popplestone, R.J. (1970) "An Experiment in Automatic Induction" Machine Intelligence 5, American Elsevier, New York.

Rabin, Michael O. (1967) "Mathematical Theory of Automata", Mathematical Aspects of Computer Science, Proc. Symp. App. Math., Amer. Math. Soc., Providence, 153-175.

Raphael, Bertram (1976) The Thinking Computer: Mind Inside Matter, W.H. Freeman, San Francisco.

Reber, Arthur S. (1967) "Implicit Learning of Artificial Grammars", J. Verbal Learning and Verbal Behavior 6, 855-863.

Reber, Arthur S. (1969) "Transfer of Syntactic Structure in Synthetic Languages", J. Exp. Psych. 81(1) 115-119.

Reber, Arthur S. (1976) "Implicit Learning of Synthetic Languages: The Role of Instructional Set", J. Exp. Psych: Human Learning and Memory 2(1) 88-94.

Reeker, Larry H. (1976) "The Computational Study of Language Acquisition" Advances in Computers, v. 15, Rubinoff and Yovits (eds.) Academic Press, New York.

Reilly, Francis E. (1970) "Charles Peirce's Theory of Scientific Method" Fordham UP, New York.

Rosenfeld, A. and J.W. Snievely,Jr., "Bounds on the Complexity of Grammars" in Frontiers of Pattern Recognition, S. Watanabe (ed), Academic Press, New York.

Rine, D.C.(1976) "Large Systems and their Regular Expressions: An Approach to Pattern Recognition", unpublished manuscript.

Rubinoff, Morris and Marshall Yovits (1975), Advances in Computers, v. 13, Academic Press, New York.

Sakoda, William (1975) "An Application of the Theory of Complexity to Inductive Inference" unpublished manuscript.

Salomaa, Arto (1969) Theory of Automata, Pergammon Press, Oxford.

Salomaa, Arto (1971) "The Generative Capacity of Transformational Grammars of Ginsburgh and Partee" Information and Control 18, 227-232.

Sapir, Edward (1921) Language, Harcourt, Brace and World, New York.

Savitch, Walter J. (1973) "How to Make Arbitrary Grammars Look Like Context-free Grammars", SIAM J. Computing 2(3), 174-182.

Schaffner, Mario R. (1971) "A Computer Modeled After an Automaton" Computers and Automata, Jerome Fox (ed.) Polytechnic Press, New York, 635-650.

Scheffler, Israel (1958) "Inductive inference: a new approach" Science, 127(3291), 177-181.

Scheffler, Israel (1963) The Anatomy of Inquiry, Knopf, New York.

Sgall, Petr, etal.(1969) A Functional Approach to Syntax in Generative Description of Language, American Elsevier, New York.

Shamir, Emil (1962) "A Remark on Discovery Algorithms for Grammars", Inf. and Control 5, 246-251.

Shannon, Claude and J. McCarthy (1956) Automata Studies, Princeton University Press, Princeton.

Shapiro, Stuart C. (1971) "A Net Structure for Semantic Information Storage, Deduction and Retrieval", Int. Joint Conf. Artificial Intelligence (2nd), London, Xerox Univ. Microfilms, Ann Arbor, 1975.

Siklossy, Laurent (1970) "On the Evolution of Artificial Intelligence" Inf. Sciences, 2, 369-377.

Singleton, Jane (1974) "The Explanatory Power of Chomsky's Transformational Generative Grammar, MIND 83(331).

Slagle, James R. (1971) Artificial Intelligence: The Heuristic Programming Approach, McGraw-Hill, New York.

Smithe, Raoul N. (1973) Probabilistic Performance Models of Language, Mouton, The Hague.

Snively, J.W. Jr. (1970) "Bounds on Complexity of Grammars" TR70-114 Univ. Maryland.

Solomonoff, Ray J. (1957) "An Inductive Inference Machine", IRE National Convention Record V, part 2.

Solomonoff, Ray J. (1958) "The Mechanization of Linguistic Learning", Deuxieme Congres Inter. de Cybern. Actes. Assoc. Inter. de Cybern., Namur.

Solomonoff, Ray J. (1959) "A Progress Report on Machines to Learn to Translate Languages and Retrieve Information", Advances in Documentation and Library Sciences, v. 3(2), Interscience, New York.

Solomonoff, Ray J. (1959) "A New Method for Discovering the Grammars of Phrase Structure Languages", Information Processing, Proc. Int. Conf. on Inf. Processing, UNESCO, Paris.

Solomonoff, R. J. (1962) "Training Sequences for Mechanized Induction" in Self-Organizing Systems 1962, Yovits, et.al., op.cit., 425-434.

Solomonoff, R.J. (1964) "A Formal Theory of Inductive Inference" Inf. and Control 7, 1-22, 224-254.

Solomonoff, R.J. (1966) "Some Recent Work in Artificial Intelligence", Proc. IEEE, 54(12), 1687-1697.

Solomonoff, R.J. (1967) "Inductive Inference Research Status Spring," Rockford Research Inst., Cambridge.

Solomonoff, R.J. (1975) "The Adequacy of Complexity Models of Induction", 5th Int. Conf. Logic, Methodology and Philosophy of Science, London (Canada).

Solomonoff, R.J. (1975) "Inductive Inference Theory - A Unified Approach to Problems in Pattern Recognition and Artificial Intelligence", 4th Int. Joint Conf. on Artificial Intelligence, Tbilisi.

Spanier, E. (1969) "Grammars and Languages", Am. Math. Monthly 76(4).

Tal', A.A. (1964) "Questionnaire Language and Abstract Synthesis of Minimal Sequential Machines", NASA Trans. TT F-260, Avtamatika i Telemekhanika 25(6), 946-962.

Thompson, Richard A. (1976) "Language Correction Using Probabilistic Grammars", IEEE Trans. Computers C-25(3).

Tou, Julius T. (ed.)(1968) Applied Automata Theory, Academic Press, New York.

Tou, Julius T. (ed.)(1969) Software Engineering: *COINS III*, Academic Press, New York.

Tou, Julius T. and Rafael C. Gonzalez (1974) Pattern Recognition Principles, Addison-Wesley, Reading.

Tsetlin, M.L. (1973) Automaton Theory and Modeling of Biological Systems, Trans. by Scitran, Academic Press, N.Y.

Turing, A.M. (1936) "On Computable Numbers, With an Application to the Entscheidungsproblem", Proc. London Math. Soc. Ser. 2-42, 230-265.

Uhr, Leonard (1973) Pattern Recognition, Learning, and Thought: Computer-Programmed Models of Higher Mental Process, Prentice-Hall, Englewood Cliffs.

Watanabe, Satosi (1965) "A Mathematical Explication of Inductive Inference" Coll. Fnd. Math. Mach. and Appl., L. Kalmar (ed.) Ak. Kiado, Budapest.

Wood, D. (1970) "Bibliography of Formal Language Theory and Automata Theory", Computing Reviews 11(7), 417-430.

Wolff, Peter H. (1967) "Cognitive Considerations for a Psychoanalytic Theory of Language Acquisition" (Psychological Issues 5(2-3)), Motives and Thoughts: Psychoanalytic Essays in Honor of David Rapaport, Robert R. Holt (ed): International Univ. Press, New York.

Yovits, Marshall, G.T. Jacobi, and G.D. Goldstein (eds). (1962), Self-Organizing Systems - 1962, Spartan, Washington.

The below listing is the official distribution list for the technical, annual, and final reports for Contract N00014-77-C-0248.

Defense Documentation Center Cameron Station Alexandria, VA 22314	12 copies
Office of Naval Research Information Systems Program Code 437 Arlington, VA 22217	2 copies
Office of Naval Research Code 102IP Arlington, VA 22217	6 copies
Office of Naval Research Code 200 Arlington, VA 22217	1 copy
Office of Naval Research Code 455 Arlington, VA 22217	1 copy
Office of Naval Research Code 458 Arlington, VA 22217	1 copy
Office of Naval Research Branch Office, Boston 495 Summer Street Boston, MA 02210	1 copy

Office of Naval Research
Branch Office, Chicago
536 South Clark Street
Chicago, IL 60605
1 copy

Office of Naval Research
Branch Office, Pasadena
1030 East Green Street
Pasadena, CA 91106
1 copy

Office of Naval Research
New York Area Office
715 Broadway - 5th Floor
New York, NY 10003
1 copy

Naval Research Laboratory
Technical Information Division, Code 2627
Washington, D.C. 20375
6 copies

Dr. A. L. Slafkosky
Scientific Advisor
Commandant of the Marine Corps (Code RD-1)
Washington, D.C. 20380
1 copy

Naval Ocean Systems Center
Advanced Software Technology Division
Code 5200
San Diego, CA 92152
1 copy

Mr. E. H. Gleissner
Naval Ship Research & Development Center
Computation and Mathematics Department
Bethesda, MD 20084
1 copy

Captain Grace M. Hopper
NAICOM/MIS Planning Branch (OP-916D)
Office of Chief of Naval Operations
Washington, D.C. 20350
1 copy

Mr. Kin B. Thompson
Technical Director
Information Systems Division (OP-942)
Office of Chief of Naval Operations
Washington, D.C. 20350
1 copy